Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра інфокомунікаційних систем і технологій

Бакалаврська дипломна робота на тему:

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ОЦІНЮВАННЯ РОБОТИ ЦИФРОВОГО ПРИЙМАЧА ПРОГРАМНО-КЕРОВАНОЇ ІНФОКОМУНІКАЦІЙНОЇ СИСТЕМИ

Виконав: студент 2-го курсу, групи ПЗТ-22мс
спеціальності 172 – Телекомунікації та радіотехніка
_____ 11.06 Муревич Р. В.

Керівник: к.т.н., доц., доцент каф. ІКСТ
_____ Бортник С.Г.
«11» 06 2024 р.

Рецензент: д.т.н., проф., професор каф. ІРТС
_____ Семенов А.О.
«11» 06 2024 р.

Допущено до захисту

Завідувач кафедри ІКСТ

_____ д.т.н., проф. Кичак В.М.

«11» 06 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра інфокомунікаційних систем та технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 17 – Електроніка та телекомунікації
(шифр і назва)
Спеціальність 172 – Телекомунікації та радіотехніка
(шифр і назва)
Освітня програма – Програмне забезпечення телекомунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКСТ
д.т.н., професор В.М. Кичак
«06» 05 2024 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Муревичу Роману Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи
керівник роботи Бортник Сергій Геннадійович, канд. техн. наук, доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ВНТУ від "11" березня 2024 року № 80
2. Строк подання студентом роботи: 07 червня 2024 року

3. Вихідні дані до роботи: тип інфокомунікаційної системи – безпровідна; режим роботи інфокомунікаційної системи - багатоканальна; тип стандарту зв'язку – супутниковий Ku діапазону; сумарна швидкість інформаційних потоків - Ethernet 20 Мбіт/с; імовірність помилки в радіотракті - 10^{-6}; формат модуляції приймача - багатопозиційна фазова маніпуляція з переходом на комбіновану; типи кодування сигналів – двоетапне із виправленням помилок; метод мультиплексування – частотний з переходом в комбінований; мобільність користувача – до 70 км/год; площа покриття інфокомунікаційної системи – згідно технічних параметрів телекомунікаційного обладнання базової станції супутникового зв'язку в Ku діапазоні; режим роботи приймача – адаптивний зі зміною формату модуляції та типу кодування сигналів.
4. Зміст розрахунково-пояснювальної записки роботи (перелік питань, які потрібно розробити): вступ, дослідження програмних моделей цифрових приймачів інфокомунікаційних систем, аналіз моделей цифрових приймачів для фазової маніпуляції сигналів, моделювання цифрових приймачів для частотної маніпуляції сигналів, програмна оптимізація цифрових приймачів, дослідження моделей цифрових модемів мобільних систем, охорона праці.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

структурна схема субоптимального приймача DBPSK сигналів; структурна модель оптимального некогерентного приймача DBPSK сигналів; диференційний демодулятор базової смуги для π/4-DQPSK; демодулятор MSK з використанням підходу комплексного представлення; імітаційна модель для когерентного та некогерентного детектування MFSK сигналів; лістинг програми…….

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Спеціальна частина	Бортник С.Г., доцент кафедри ІКСТ	06.05.2024	11.06.2024
Охорона праці	Дембицька С.В. проф. каф. БЖДЛБ		

7. Дата видачі завдання «06» травня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Формування та затвердження теми та індивідуального завдання бакалаврської дипломної роботи (БДР)	06.05.2024р.	
2	Виконання спеціальної частини БДР. Перший рубіжний контроль виконання БДР	20.05.2024р.	
3	Виконання спеціальної частини БДР. Другий рубіжний контроль виконання БДР	31.05.2024р.	
4	Виконання розділу «Охорона праці»	04.06.2024р.	
5	Нормоконтроль БДР	05.06.2024р.	
6	Попередній захист БДР	06.06.2024р.	
7	Рецензування БДР	07.06.2024р.	
8	Захист БДР	12.06.2024р.	

Студент _____ (підпис) Муревич Р.В.

Керівник роботи _____ (підпис) Бортник С.Г.

АНОТАЦІЯ

Розробка програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи. Бакалаврська дипломна робота / Р. В. Муревич – Вінниця: ВНТУ, 2024. – 116 с., 23 рис., 12 табл., 24 – бібл. – українською мовою.

Метою даної роботи є вдосконалення та оптимізація цифрових приймачів інфокомунікаційних систем шляхом дослідження їх продуктивності, ефективності та надійності в різних умовах та за різних обставин використання. Виконано дослідження програмних моделей цифрових приймачів інфокомунікаційних систем для симуляції та аналізу різних аспектів їхньої роботи. Здійснено аналіз моделей цифрових приймачів для фазової маніпуляції сигналів в різних умовах та сценаріях з метою вдосконалення їхньої ефективності та надійності. Виконано моделювання цифрових приймачів для частотної маніпуляції сигналів, що дозволяє аналізувати та вдосконалювати роботу цифрових приймачів FSK для забезпечення їхньої ефективності та надійності. Реалізовано програмну оптимізацію цифрових приймачів для досягнення їх оптимальної продуктивності у програмних додатках. Здійснено дослідження моделей цифрових модемів мобільних систем для вдосконалення їх роботи в мобільних системах і покращення їхньої відповідності вимогам сучасних комунікаційних потреб. Розглянуто питання з охорони праці.

Ключові слова: цифровий приймач інфокомунікаційної системи; комп’терна модель цифрового модему; спектральна густина потужності; комп’ютерне моделювання.

ABSTRACT

Development of a software application for evaluating the operation of a digital receiver of a software-controlled infocommunication system. Bachelor's thesis / R. V. Murevych - Vinnytsia: VNTU, 2024 - 116 p., 23 figs., 12 tables, 24 bibliography - in Ukrainian.

The aim of this work is to improve and optimize digital receivers of infocommunication systems by studying their performance, efficiency and reliability in different conditions and under different circumstances of use. A study of software models of digital receivers of infocommunication systems for simulation and analysis of various aspects of their operation has been carried out. The analysis of models of digital receivers for phase manipulation of signals in different conditions and scenarios is carried out in order to improve their efficiency and reliability. The modeling of digital receivers for frequency manipulation of signals was performed, which allows to analyze and improve the operation of digital FSK receivers to ensure their efficiency and reliability. Software optimization of digital receivers was implemented to achieve their optimal performance in software applications. A study of models of digital modems for mobile systems was carried out to improve their performance in mobile systems and improve their compliance with the requirements of modern communication needs. The issues of labor protection are considered.

Keywords: digital receiver of an infocommunication system; computer model of a digital modem; power spectral density; computer modeling.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 6
ВСТУП 8
1 ПРОГРАМНІ МОДЕЛІ ЦИФРОВИХ ПРИЙМАЧІВ ІНФОКОМУНІКАЦІЙНИХ СИСТЕМ 11
1.1 Приймач BPSK сигналів 11
1.2 Наскрізне симуляційне моделювання 15
1.3 Когерентне детектування диференційно кодованого BPSK сигналу 18
1.4 Диференціальна BPSK маніпуляція сигналів в безпровідних системах 28
1.5 Субоптимальний приймач для DBPSK сигналів 30
1.6 Оптимальний некогерентний приймач для диференціальної BPSK модуляції 32
1.7 Висновки до розділу 1 41
2 МОДЕЛІ ЦИФРОВИХ ПРИЙМАЧІВ ФАЗОВОЇ МАНІПУЛЯЦІЇ СИГНАЛІВ 43
2.1 Приймач QPSK сигналів 43
2.2 Формат модуляції π/4-QPSK 45
2.3 Демодулятор MSK сигналів 49
2.4 Демодулятор GMSK сигналів 53
2.5 Моделювання продуктивності формату модуляції GMSK 59
2.6 Висновки до розділу 2 61
3 МОДЕЛЮВАННЯ ЦИФРОВИХ ПРИЙМАЧІВ FSK СИГНАЛІВ 62
3.1 Когерентний демодулятор 62
3.2 Некогерентний демодулятор 66
3.3 Моделювання продуктивності демодуляції сигналу BFSK через канал AWGN 68
3.4 Висновки до розділу 3 72
4 ПРОГРАМНА ОПТИМІЗАЦІЯ ЦИФРОВИХ ПРИЙМАЧІВ 73

4.1 Оптимальний детектор на площині IQ з мінімальною евклідовою відстанню 73
4.2 М-розрядний модем з частотною маніпуляцією 79
4.3 Детектування M-FSK сигналів 83
4.4 Висновки до розділу 4 87
5 МОДЕЛІ ЦИФРОВИХ МОДЕМІВ МОБІЛЬНИХ СИСТЕМ 90
5.1 Програмна реалізація цифрових модемів 90
5.2 Програмне визначення продуктивності інфокомунікаційної системи 97
5.3 Висновки до розділу 5 99
6 ОХОРОНА ПРАЦІ 100
6.1 Технічні рішення щодо безпечного виконання роботи 100
6.2 Технічні рішення з гігієни праці та виробничої санітарії 103
6.2.1 Мікроклімат 103
6.2.2 Склад повітря робочої зони 104
6.2.3 Виробниче освітлення 105
6.2.4 Виробничий шум 106
6.2.5 Виробничі випромінювання 107
6.3 Пожежна безпека 108
6.3.1 Технічні рішення системи запобігання пожежі 109
6.3.2 Технічні рішення системи протипожежного захисту 110
ВИСНОВКИ 112
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 114
ДОДАТКИ 117
Додаток А Ілюстративна частина 118
Додаток Б Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень 123

ПЕРЕЛІК СКОРОЧЕНЬ

3GPP - Проект партнерства третього покоління

6LoWPAN - IPv6 через WPAN з низькою потужністю

A- CSCF - Контролер границі сеансу доступу

ADSL - Асиметрична цифрова абонентська лінія

AM - Амплітудна модуляція

AMPS - Розширена система мобільного зв'язку

AMQP - протокол розширеної черги повідомлень

ANN - Штучна нейронна мережа

ASK - Амплітудна маніпуляція

ATM - Асинхронний режим передачі

AuC - Центр автентифікації

BBU - Блок базового діапазону

BER - Бітовий коефіцієнт помилок

BGP - Протокол граничного шлюзу

BICN - Базова мережа, що не залежить від носія

BSC - Контролер базової станції

BSS - Підсистема базової станції

BTS - Базова приймально-передавальна станція

CA - Агрегація несучих

CDM - мультиплексування з кодовим поділом каналів

CDMA - множинний доступ з кодовим поділом каналів

CDR - Детальний запис виклику

CINR - відношення несуча/завада + шум

CIR - Канальна імпульсна характеристика

CNN - Згорткова нейронна мережа

CoAP - протокол обмежених додатків

CPC - Безперервне пакетне з'єднання

C- RAN - хмарна/централізована мережа радіодоступу

CSCP - Функція управління сеансом зв'язку

DAS - розподілена антенна система

DC - подвійне підключення

DDoS - розподілена відмова в обслуговуванні

DDS - служба розподілу даних

DNS - система доменних імен

DPWS - Профіль пристроїв для веб-сервісів

D- RAN - Розподілена мережа радіодоступу

DSCP - Диференційована кодова точка обслуговування

DTLS - Безпека транспортного рівня дейтаграм

EDGE - Розширені швидкості передачі даних для GSM Evolution

EIR - Ідентифікаційний реєстр обладнання

eMBB - Розширений мобільний широкосмуговий зв'язок

EPC - вдосконалене пакетне ядро

EV- DO - Еволюція з оптимізацією даних

FDD - Дуплексування з частотним розділенням каналів

FDM - мультиплексування з частотним розділенням каналів

FDMA - множинний доступ з частотним розділенням каналів

FM - частотна модуляція

FSK - частотна маніпуляція

FTAM - Передача, доступ і керування файлами

FTP - протокол передачі файлів

GAN - Генеративна змагальна мережа

GEO - Геосинхронна екваторіальна орбіта

GERAN - гранична мережа радіодоступу GSM

GGSN - Шлюзовий вузол підтримки GPRS

GMSC - Центр мобільної комутації шлюзу

GPOS - Операційна система загального призначення

GPRS - Загальна служба пакетного радіозв'язку

GSM - Глобальна система мобільного зв'язку

ВСТУП

Актуальність теми. Розробка програмного додатку для оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи може бути важливою для забезпечення якості та ефективності цієї системи. При цьому, спочатку необхідно чітко визначити, які саме функції повинен виконувати додаток. Це може включати в себе тестування приймача з різними типами сигналів, вимірювання параметрів, таких як сигнал-шум (SNR) або коефіцієнт прийому, а також аналіз використання каналу передачі даних. Розроблений додаток повинен мати зручний інтерфейс користувача, який дозволяв би легко виконувати необхідні дії, такі як налаштування тестів, виконання вимірювань та аналіз результатів. Для обробки сигналів можна використовувати бібліотеки Python, такі як NumPy та SciPy. Збір та обробка даних може бути виконана за допомогою пакету pandas. Для візуалізації результатів можна використовувати бібліотеки Matplotlib або Seaborn. Графіки і діаграми можуть допомогти користувачам краще зрозуміти отримані дані. Також, важливо виконати тестування додатку на різних сценаріях роботи, а також на різних конфігураціях обладнання. Потім необхідно внести виправлення та покращення на основі отриманих результатів. У підсумку, необхідно підготувати документацію для користувачів, яка пояснювала б використання додатку та його функціональність. Також важливо забезпечити підтримку користувачів та відповідати на їхні запитання та відгуки. Отже, розробка програмного додатку для оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи може виявитися складним завданням, але правильне планування та використання підходящих інструментів може допомогти зробити його більш ефективним і корисним [1, 2].

Аналіз останніх досліджень. Останні дослідження з розробки програмного додатку для оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи акцентують на декількох ключових аспектах [3]. Багато досліджень спрямовані на розробку програмних додатків, які дозволяють

автоматизувати процес тестування цифрових приймачів. Це дозволяє ефективно проводити тести з різними параметрами сигналу та умовами каналу передачі даних. Дослідники зосереджують увагу на розробці методів для вимірювання ключових параметрів цифрових приймачів, таких як чутливість, динамічний діапазон, SNR, BER тощо [4]. Важливим аспектом є аналіз впливу різноманітних шумів та спотворень на роботу цифрового приймача. Деякі дослідження фокусуються на розробці методів компенсації цих негативних ефектів [5].

Інші дослідження ставлять за мету оптимізацію алгоритмів обробки сигналу в цифрових приймачах з метою підвищення їхньої продуктивності та ефективності [6]. Деякі дослідження досліджують використання методів машинного навчання та штучного інтелекту для аналізу та вдосконалення роботи цифрових приймачів [7]. Деякі дослідження спрямовані на розширення функціональності програмних додатків для оцінювання роботи цифрових приймачів, включаючи підтримку різних стандартів зв'язку та інтерфейсів [8]. Вказані напрямки досліджень вказують на постійний інтерес до вдосконалення та розвитку програмного забезпечення для оцінювання роботи цифрових приймачів у програмно-керованих інфокомунікаційних системах [3-5].

Знання цих концепцій дозволяє інженерам та дослідникам аналізувати, розробляти та оптимізувати системи зв'язку для досягнення кращої ефективності та якості передачі даних.

Мета та постановка задачі. Метою даної бакалаврської дипломної роботи є вдосконалення та оптимізація цифрових приймачів інфокомунікаційних систем шляхом дослідження їх продуктивності, ефективності та надійності в різних умовах та за різних обставин використання.

Отже, вдосконалення та оптимізація цифрових приймачів інфокомунікаційних систем через дослідження їх продуктивності, ефективності та надійності в різних умовах є важливим завданням.

Задачами бакалаврської дипломної роботи є:

- дослідження програмних моделей цифрових приймачів інфокомунікаційних систем для симуляції та аналізу різних аспектів їхньої роботи;

- аналіз моделей цифрових приймачів для фазової маніпуляції сигналів в різних умовах та сценаріях з метою вдосконалення їхньої ефективності та надійності;

- моделювання цифрових приймачів для частотної маніпуляції сигналів, що дозволяє аналізувати та вдосконалювати роботу цифрових приймачів FSK для забезпечення їхньої ефективності та надійності;

- програмна оптимізація цифрових приймачів для досягнення їх оптимальної продуктивності у програмних додатках;

- дослідження моделей цифрових модемів мобільних систем для вдосконалення їх роботи в мобільних системах і покращення їхньої відповідності вимогам сучасних комунікаційних потреб.

- вивчення питань з охорони праці.

Апробація результатів роботи. Основні ідеї роботи доповідались і обговорювались на науковій конференції ВНТУ у 2024 році.

1 ПРОГРАМНІ МОДЕЛІ ЦИФРОВИХ ПРИЙМАЧІВ ІНФОКОМУНІКАЦІЙНИХ СИСТЕМ

Програмні моделі роботи цифрових приймачів інфокомунікаційних систем є важливим інструментом для аналізу, моделювання та вдосконалення цих систем. Кожна з цих програмних моделей може бути розроблена, імплементована та використана для аналізу та вдосконалення роботи цифрових приймачів інфокомунікаційних систем. Вони допомагають розробникам та інженерам отримувати глибше розуміння процесів, що відбуваються в системі, та здійснювати ефективніше вдосконалення та оптимізацію її роботи.

1.1 Приймач BPSK сигналів

Для реалізації приймача використовується когерентний детектор кореляційного типу, показаний на рисунку 1.1. Так, при моделюванні системи когерентного детектування можна припустити, що фаза несучої була відновлена, і використовувати згенеровану опорну частоту на приймачі. Це може спростити моделювання та аналіз системи, зосередивши увагу на інших аспектах дослідження, наприклад, алгоритмах демодуляції або впливі каналу передачі [1].

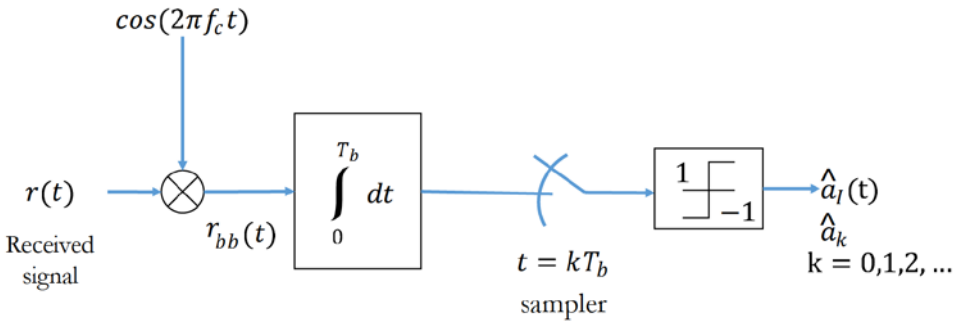

Рисунок 1.1 - Структурна модель для когерентного детектування BPSK сигналів

Когерентне детектування BPSK (Binary Phase Shift Keying) зазвичай використовує кореляційний метод для визначення переданих бітів. Основна ідея полягає в порівнянні фази отриманого сигналу з фазою несучої хвилі, що була використана для модуляції. Наведемо алгоритм когерентного детектування BPSK:

Отримайте вхідний сигнал BPSK, який складається з послідовності бітів, що були модульовані за допомогою BPSK. Визначте фазу несучої хвилі, яка використовувалася при передачі сигналу. Це може бути зроблено шляхом синхронізації з отриманим сигналом або за допомогою передачі спеціального синхронізаційного сигналу разом з даними. Переведіть отриманий сигнал у комплексну форму, яка включає інформацію про амплітуду та фазу. Здійсніть комплексний зсув фази, використовуючи фазу несучої хвилі, щоб відновити фазу сигналу. Порівняйте фазу відновленого сигналу з опорною фазою, яка відповідає передачі біта 0. Якщо фаза сигналу близька до опорної, то вважається, що переданий біт - 0; в іншому випадку, біт вважається 1. Підтвердіть передачу біта та продовжуйте обробку даних, переходячи до наступних бітів у послідовності.

Цей процес дозволяє ефективно виявляти передані біти у сигналі BPSK, використовуючи кореляційний метод та враховуючи фазову інформацію сигналу. Когерентне детектування забезпечує стійкість до фазових зміщень та шуму, що допомагає досягти високої надійності в передачі даних.

Петля Костаса та петля фазової автопідстройки частоти (ФАПЧ) є ефективними методами для відновлення фази несучої хвилі в системі когерентного детектування. Вони дозволяють утримувати правильну фазу та частоту несучої хвилі навіть у випадку її змін, що допомагає забезпечити стабільну та надійну роботу системи. Проте, якщо ці методи не використовуються у моделі, можна просто припустити, що фаза несучої відновлена і використовувати згенеровану опорну частоту для подальшої обробки сигналу. Це дозволить уникнути складнощів з моделюванням фазового відновлення та сконцентрувати увагу на інших аспектах дослідження системи когерентного детектування.

У когерентному приймачі прийнятий сигнал множиться на сигнал опорної частоти з блоків відновлення несучої, таких як PLL або петля Костаса. Тут припускається, що PLL/Costas петля присутня і вихід повністю синхронізований. Перемножений вихід інтегрується протягом одного бітового періоду за допомогою інтегратора.

Пороговий детектор приймає рішення щодо кожного інтегрованого біта на основі порогового значення. Оскільки в передавачі використовувався формат сигналу NRZ, поріг для детектора буде встановлено на 0. Функція bpsk demod реалізує приймач з базовою смугою BPSK, як показано на рисунку 1.1. Щоб використовувати цю функцію для моделювання форми сигналу, спочатку потрібно перетворити отриману форму сигналу до базової смуги, а потім викликати функцію. Отже, використаємо функцію `bpsk_demod`, щоб реалізувати когерентний приймач BPSK з пороговим детектором. Ця функція припускає, що вихідний сигнал вже знаходиться у базовій смузі BPSK, тобто сигнал вже був перемножений на опорну частоту та інтегрований протягом одного бітового періоду [2].

```
import numpy as np

def bpsk_demod(received_signal):
    # Визначення порогового значення
    threshold = 0

    # Пороговий детектор
    demodulated_bits = (received_signal > threshold).astype(int)

    return demodulated_bits

# Передані дані
received_signal = np.array([...])  # Отриманий сигнал у базовій смузі BPSK
```

```
# Демодуляція сигналу
demodulated_bits = bpsk_demod(received_signal)

# Вивід результату демодуляції
print("Demodulated bits:", demodulated_bits)
```

Важливо врахувати, що цей підхід передбачає, що отриманий сигнал вже знаходиться у базовій смузі BPSK та проходить крізь пороговий детектор. Якщо отриманий сигнал ще не перетворено у базову смугу BPSK, це потрібно зробити перед викликом функції `bpsk_demod`.

Базова смуга BPSK (Binary Phase Shift Keying) представляє собою сигнал, який має частотний спектр інформаційного сигналу та дві основні частоти, що визначаються амплітудою і фазою несучої хвилі. У випадку BPSK ці частоти зазвичай розміщуються навколо нульової частоти, тому що BPSK модулюється фазою сигналу, а не зміною амплітуди. Для визначення базової смуги BPSK, необхідно виконати наступні кроки. Використовуйте метод модуляції BPSK для перетворення бітової послідовності на сигнал. У BPSK кожен біт 0 представляється сигналом з фазою 0 градусів, а кожен біт 1 - сигналом з фазою 180 градусів. Після модуляції отриманий сигнал може бути зсунутий у частотній області, щоб його центральна частота знаходилася навколо нульової частоти. Це можна зробити за допомогою зсуву Фур'є-перетворення. У випадку необхідності застосуйте фільтрацію для обмеження спектру сигналу до потрібної полоси частот. В цілому, базова смуга BPSK буде мати діапазон частот, який зосереджений навколо нульової частоти, з двома основними піковими частотами, відокремленими на відстані, яка відповідає бітовій швидкості сигналу.

```
def bpsk_demod(r_bb,L):
    """
    Function to demodulate a BPSK (baseband) signal
    Parameters:
        r_bb : received signal at the receiver front end (baseband)
        L : oversampling factor (Tsym/Ts)
    Returns:
        ak_hat : detected/estimated binary stream
    """
    x = np.real(r_bb) # I arm
    x = np.convolve(x,np.ones(L)) # integrate for Tb duration (L samples)
    x = x[L-1:-1:L] # I arm - sample at every L
    ak_hat = (x > 0).transpose() # threshold detector
    return ak_hat
```

1.2 Наскрізне симуляційне моделювання

Наскрізне симуляційне моделювання (end-to-end simulation) - це метод моделювання, який відтворює весь процес або систему з початку до кінця, включаючи всі компоненти та їх взаємодію. Цей підхід дозволяє оцінити роботу системи в її повному контексті та з врахуванням усіх можливих впливів і факторів.

У наскрізному моделюванні моделюються всі компоненти та елементи системи, включаючи джерела сигналу, канали передачі, методи модуляції/демодуляції, приймачі, а також будь-які обробки сигналу та інші обчислювальні блоки. Важливо врахувати взаємодію між різними компонентами системи. Це може включати передачу сигналу через канал, зміну параметрів сигналу під час передачі, вплив шумів та спотворень, а також реакцію приймача на отриманий сигнал. Після завершення симуляції важливо проаналізувати отримані результати. Це може включати оцінку якості передачі сигналу, продуктивності системи, рівня помилок та інших метрик ефективності.

Наскрізне симуляційне моделювання дозволяє відтворити реальні умови роботи системи, такі як різні канали передачі, рівні шуму, спотворення та інші чинники, які можуть впливати на її функціонування. Симуляційне моделювання дозволяє вносити зміни та покращення в систему без реального впровадження змін, що дозволяє оптимізувати її роботу та вибрати найбільш ефективні

параметри. Загалом, наскрізне симуляційне моделювання є потужним інструментом для аналізу та вдосконалення роботи різноманітних систем, включаючи телекомунікаційні, мережеві та інші.

Далі наведено повне моделювання форми сигналу для наскрізної передачі інформації з використанням BPSK модуляції. Моделювання включає: генерацію випадкових бітів повідомлення, модуляцію їх за допомогою BPSK-модуляції, додавання шуму AWGN відповідно до обраного співвідношення сигнал/шум і демодуляцію зашумленого сигналу за допомогою когерентного приймача. Отримані графіки форми сигналу показано на рисунку 1.2. [3]

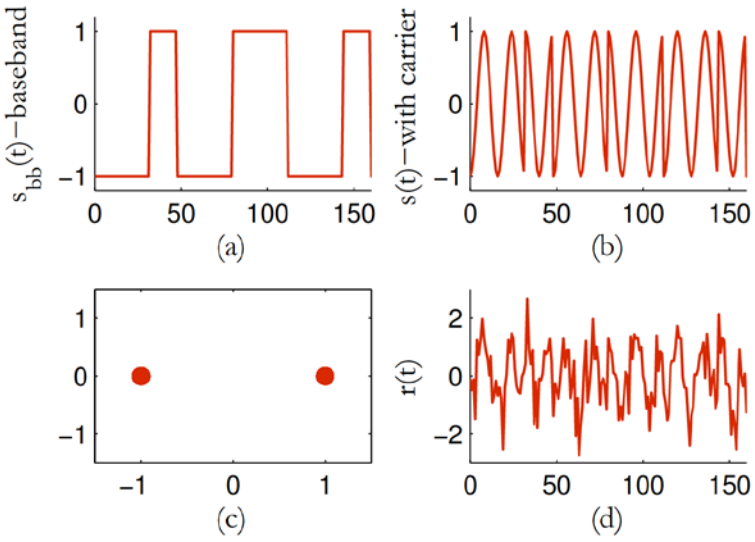

Рисунок 1.2 - (a) Сигнал BPSK базової смуги, (b) переданий сигнал BPSK - з несучою, (c) сузір'я на передавачі та (d) прийнятий сигнал з шумом AWGN

Моделювання продуктивності для комбінації передавача/приймача з BPSK також закодовано в програмі, наведеній нижче. Отримані криві продуктивності будуть такими ж, як і криві, отримані за допомогою методу моделювання еквівалентної комплексної смуги пропускання [4].

Продуктивність системи BPSK може бути оцінена за допомогою симуляції форми сигналу. Наведемо послідовність основних кроків, які можна виконати для такої оцінки. Згенеруйте випадкову послідовність бітів, яка буде

передаватися за допомогою системи BPSK. Ці дані можуть бути згенеровані згідно з ваших потреб або використовувати реальні дані, якщо такі є доступні. Застосуйте метод модуляції BPSK до послідовності бітів, щоб створити сигнал BPSK. Це можна зробити шляхом зміни фази несучої хвилі відповідно до значень бітів. Створіть шумовий сигнал, який симулює шум та спотворення, які можуть виникати в каналі передачі. Додавання цього шуму дозволяє оцінити роботу системи в умовах реального каналу зв'язку. Проведіть демодуляцію та декодування отриманого сигналу BPSK, щоб відновити передані дані. Використовуйте відповідні алгоритми демодуляції та декодування, які відповідають методу модуляції. Оцініть продуктивність системи, використовуючи метрики, такі як швидкість передачі, рівень помилок, співвідношення сигнал/шум та інші відповідні показники ефективності. Проаналізуйте отримані результати, щоб зрозуміти, які чинники впливають на продуктивність системи та як можна її покращити. Це може включати вплив шуму, спотворень каналу, рівня модуляції та інші параметри.

Вказані кроки дозволяють провести дослідження та оцінку продуктивності системи BPSK з використанням симуляції форми сигналу. Важливо враховувати усі можливі впливи та фактори, які можуть впливати на роботу системи, для отримання достовірних результатів [5].

```
#Execute in Python3: exec(open("chapter_2/bpsk.py").read())
import numpy as np #for numerical computing
import matplotlib.pyplot as plt #for plotting functions
from DigiCommPy.passband_modulations import bpsk_mod, bpsk_demod
from DigiCommPy.channels import awgn
from scipy.special import erfc

N=100000 # Number of symbols to transmit
EbN0dB = np.arange(start=-4,stop = 11,step = 2) # Eb/N0 range in dB for simulation
L=16 # oversampling factor,L=Tb/Ts(Tb=bit period,Ts=sampling period)

# if a carrier is used, use L = Fs/Fc, where Fs >> 2xFc
Fc=800 # carrier frequency
Fs=L*Fc # sampling frequency
BER = np.zeros(len(EbN0dB)) # for BER values for each Eb/N0
ak = np.random.randint(2, size=N) # uniform random symbols from 0's and 1's
(s_bb,t)= bpsk_mod(ak,L) # BPSK modulation(waveform) - baseband
s = s_bb*np.cos(2*np.pi*Fc*t/Fs) # with carrier
```

```
# Waveforms at the transmitter
fig1, axs = plt.subplots(2, 2)
axs[0, 0].plot(t,s_bb) # baseband wfm zoomed to first 10 bits
axs[0, 0].set_xlabel('t(s)');axs[0, 1].set_ylabel(r'$s_{bb}(t)$-baseband')
axs[0, 1].plot(t,s) # transmitted wfm zoomed to first 10 bits
axs[0, 1].set_xlabel('t(s)');axs[0, 1].set_ylabel('s(t)-with carrier')
axs[0, 0].set_xlim(0,10*L);axs[0, 1].set_xlim(0,10*L)
#signal constellation at transmitter
axs[1, 0].plot(np.real(s_bb),np.imag(s_bb),'o')
axs[1, 0].set_xlim(-1.5,1.5);axs[1, 0].set_ylim(-1.5,1.5)

for i,EbN0 in enumerate(EbN0dB):
    # Compute and add AWGN noise
    r = awgn(s,EbN0,L) # refer Chapter section 4.1

    r_bb = r*np.cos(2*np.pi*Fc*t/Fs) # recovered baseband signal
    ak_hat = bpsk_demod(r_bb,L) # baseband correlation demodulator
    BER[i] = np.sum(ak !=ak_hat)/N # Bit Error Rate Computation

    # Received signal waveform zoomed to first 10 bits
    axs[1, 1].plot(t,r) # received signal (with noise)
    axs[1, 1].set_xlabel('t(s)');axs[1, 1].set_ylabel('r(t)')
    axs[1, 1].set_xlim(0,10*L)
#------Theoretical Bit/Symbol Error Rates-------------
theoreticalBER = 0.5*erfc(np.sqrt(10**(EbN0dB/10))) # Theoretical bit error rate
#-------------Plots---------------------------
fig2, ax1 = plt.subplots(nrows=1,ncols = 1)
ax1.semilogy(EbN0dB,BER,'k*',label='Simulated') # simulated BER
ax1.semilogy(EbN0dB,theoreticalBER,'r-',label='Theoretical')
ax1.set_xlabel(r'$E_b/N_0$ (dB)')
ax1.set_ylabel(r'Probability of Bit Error - $P_b$')
ax1.set_title(['Probability of Bit Error for BPSK modulation'])
ax1.legend();fig1.show();fig2.show()
```

1.3 Когерентне детектування диференційно кодованого BPSK сигналу

Когерентне детектування диференційно кодованого BPSK (DEBPSK) використовується для демодуляції сигналів, що зашифровані за допомогою диференційного кодування перед фазовою модуляцією. Основна відмінність DEBPSK від звичайного BPSK полягає в тому, що кожен біт кодується не окремо, а відносно попереднього біта. DEBPSK використовується для зменшення впливу зміщень фази у каналі передачі. Це робить його більш стійким до змін фази, які можуть виникати під час передачі сигналу. Крім того, він зазвичай використовує менше простору для передачі інформації, оскільки кожен біт кодується відносно попереднього, що робить його ефективним для використання в умовах обмеженого каналу передачі.

Диференційне кодування може бути використане для ефективного управління фазовою неоднозначністю, що може виникати у когерентному детектуванні BPSK. Замість кодування інформації як абсолютні фази, вона кодується як різниця фаз між двома послідовними відліками.

Наведемо основні ключові кроки у використанні диференційного кодування для когерентного детектування BPSK. Замість безпосереднього кодування інформації як абсолютні фази, визначте різницю фази між двома послідовними відліками і скористайтеся цим значенням як вхід для BPSK-модулятора. Наприклад, якщо попередній відлік має фазу q1, а поточний відлік має фазу q2, то відлік буде модульований як BPSK сигнал з фазою q2 - q1. Після отримання сигналу на приймачі, визначте різницю фази між послідовними відліками сигналу. Ця різниця фази буде використовуватися для демодуляції BPSK сигналу. Наприклад, якщо фаза поточного відліку - q2, а попереднього - q1, то можна визначити, що біт 0 був переданий, якщо q2 - q1 > 0, і біт 1, якщо q2 - q1 < 0 [6].

Використання диференційного кодування дозволяє позбутися фазової невизначеності, яка може виникати при когерентному детектуванні BPSK, і зробити систему більш стійкою до фазових зміщень у каналі передачі. Це покращує якість передачі даних і дозволяє ефективно використовувати когерентне детектування для отримання правильних результатів.

Отже, з фазовою неоднозначністю можна ефективно боротися, застосовуючи диференційне кодування на вході BPSK-модулятора (рис. 1.3) і виконуючи диференційне декодування на виході когерентного демодулятора на стороні приймача (рис. 1.4). У звичайній передачі BPSK інформація кодується як абсолютні фази: q = 0 для двійкової 1 і q = 180 для двійкового 0. При диференціальному кодуванні інформація кодується як різниця фаз між двома послідовними відліками [7].

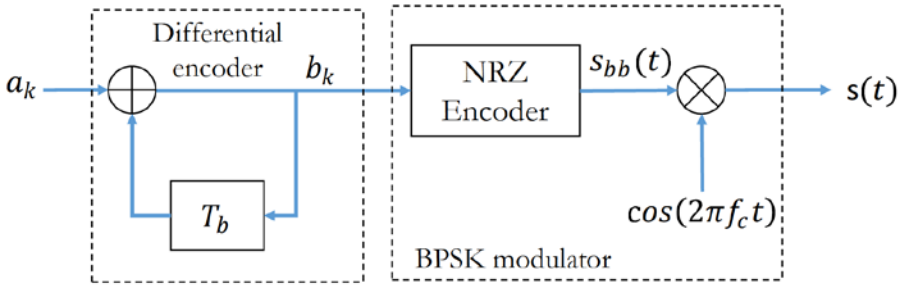

Рисунок 1.3 – Структурна схема моделі передавання з диференціальним кодуванням BPSK

Передача з диференціальним кодуванням BPSK (Differential Binary Phase Shift Keying) є одним з методів модуляції, який використовується в безпровідних комунікаційних системах для ефективної передачі даних. Основна ідея полягає в тому, що інформація кодується як різниця фаз між послідовними символами, а не як абсолютні значення фази. Це дозволяє покращити стійкість до змін фази, що можуть виникати в каналі передачі. Передача з диференціальним кодуванням BPSK забезпечує певний рівень стійкості до змін фази в каналі передачі, що робить її ефективним методом модуляції для безпровідних комунікаційних систем [8].

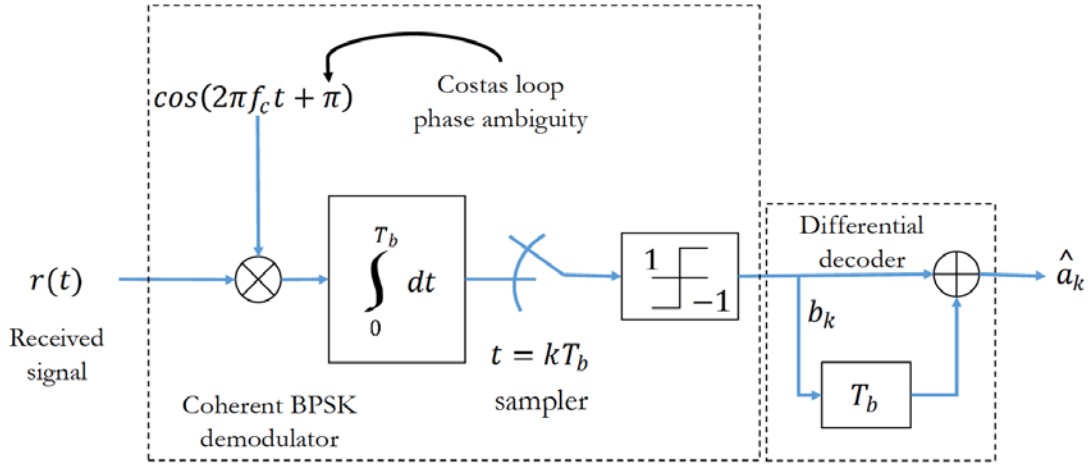

Рисунок 1.4 – Когерентне детектування диференційно кодованого BPSK сигналу

Когерентне детектування диференційно кодованого BPSK (Differential BPSK) сигналу використовується для демодуляції сигналів, які були диференційно закодовані перед фазовою модуляцією. Основна відмінність між звичайним BPSK і диференційним BPSK полягає в тому, що кожний символ кодується як різниця фази між поточним і попереднім символом, а не як абсолютна фаза. Це дозволяє знизити вплив фазової невизначеності та покращити стійкість до спотворень каналу. Когерентне детектування диференційно кодованого BPSK сигналу дозволяє знизити вплив фазової невизначеності та покращити стійкість до шуму та спотворень в каналі передачі. Це робить його ефективним методом для безпроводних комунікаційних систем, де важлива стійкість до перешкод.

Так, одним з недоліків диференційного декодування BPSK є втрата продуктивності в разі, коли попередній біт помилковий, а поточний біт правильний, або навпаки. Це відбувається через те, що диференційне декодування сприймає зміну фази як зміну біту, незалежно від того, чи був попередній біт правильним чи ні.

Щоб вирішити цей недолік, можна використовувати додаткові методи корекції помилок або модифікувати сам алгоритм демодуляції. Одним із широко використовуваних підходів є використання зворотного каналу зворотного зв'язку, такого як ARQ (Automatic Repeat reQuest) або FEC (Forward Error Correction), які дозволяють виявити та виправити помилки в переданих даних. Ці методи можуть значно знизити вплив неправильної демодуляції на продуктивність системи. Крім того, можливим рішенням є застосування адаптивних алгоритмів демодуляції, які здатні адаптувати параметри демодуляції залежно від умов каналу передачі. Це може включати різноманітні методи фільтрації та алгоритми прийняття рішень, які дозволяють ефективно вирішувати проблеми неправильної демодуляції в умовах невизначеності фази.

Нижче наведено реалізацію на мові Python імітаційної моделі сигналу для дослідженого методу.

Блоки диференціального кодування та диференціального декодування, зображені на рисунках 1.3 та 1.4), є лінійно-інваріантними в часі фільтрами. Диференціальний кодер реалізовано за допомогою цифрового фільтра типу IIR, а диференціальний декодер реалізовано за допомогою фільтра типу FIR [9].

Наведемо приклад лістингу коду Python для когерентного детектування диференційно кодованого BPSK (DEBPSK):

```
import numpy as np

def coherent_detection_DEBPSK(received_signal, sampling_frequency, carrier_frequency, symbol_rate):
    # Функція для когерентного детектування диференційно кодованого BPSK

    # Генерація несучої хвилі
    time = np.arange(0, len(received_signal) / sampling_frequency, 1 / sampling_frequency)
    carrier_wave = np.cos(2 * np.pi * carrier_frequency * time)

    # Множення отриманого сигналу на несучу хвилю
    received_signal *= carrier_wave

    # Перемноження вихідного сигналу і несучої хвилі, інтегрування протягом одного бітового періоду
    integrated_signal = np.convolve(received_signal, carrier_wave, mode='valid')

    # Пороговий детектор
    threshold = 0  # Поріг детектора
    detected_bits = np.where(integrated_signal > threshold, 1, 0)
```

```
    return detected_bits

# Приклад використання функції для детектування
received_signal = np.array([...])  # Отриманий сигнал
sampling_frequency = 100e6  # Частота дискретизації сигналу (100 МГц)
carrier_frequency = 1e6  # Частота несучої хвилі (1 МГц)
symbol_rate = 1e3  # Символьна швидкість (1 кГц)

detected_bits = coherent_detection_DEBPSK(received_signal, sampling_frequency, carrier_frequency, symbol_rate)
print("Detected bits:", detected_bits)
```

У цьому коді функція `coherent_detection_DEBPSK` приймає отриманий сигнал `received_signal`, частоту дискретизації `sampling_frequency`, частоту несучої хвилі `carrier_frequency` та символьну швидкість `symbol_rate`. Вона виконує когерентне детектування DEBPSK шляхом множення отриманого сигналу на несучу хвилю, інтегрування та порогового детектування. Цей код можна модифікувати або розширити відповідно до специфіки вашого додатку або вимог вашого проекту.

```
#Execute in Python3: exec(open("chanter_2/debpsk_coherent.py").read())
import numpy as np #for numerical computing
import matplotlib.pyplot as plt #for plotting functions
from DigiCommPy.passband_modulations import bpsk_mod, bpsk_demod
from DigiCommPy.channels import awgn
from scipy.signal import lfilter
from scipy.special import erfc

N=1000000 # Number of symbols to transmit
EbN0dB = np.arange(start=-4,stop = 11,step = 2) # Eb/N0 range in dB for simulation
L=16 # oversampling factor,L=Tb/Ts(Tb=bit period,Ts=sampling period)
# if a carrier is used, use L = Fs/Fc, where Fs >> 2xFc
Fc=800 # carrier frequency
Fs=L*Fc # sampling frequency
```

```
SER = np.zeros(len(EbN0dB)) # for SER values for each Eb/N0

ak = np.random.randint(2, size=N) # uniform random symbols from 0's and 1's
bk = lfilter([1.0],[1.0,-1.0],ak) #IIR filter for differential encoding
bk = bk%2 #XOR operation is equivalent to modulo-2

[s_bb,t]= bpsk_mod(bk,L) # BPSK modulation(waveform) - baseband
s = s_bb*np.cos(2*np.pi*Fc*t/Fs) # DEBPSK with carrier

for i,EbN0 in enumerate(EbN0dB):
    # Compute and add AWGN noise
    r = awgn(s,EbN0,L) # refer Chapter section 4.1

    phaseAmbiguity=np.pi # 180* phase ambiguity of Costas loop
    r_bb=r*np.cos(2*np.pi*Fc*t/Fs+phaseAmbiguity) # recovered signal
    b_hat=bpsk_demod(r_bb,L) # baseband correlation type demodulator
    a_hat=lfilter([1.0,1.0],[1.0],b_hat) # FIR for differential decoding
    a_hat= a_hat % 2 # binary messages, therefore modulo-2
    SER[i] = np.sum(ak !=a_hat)/N #Symbol Error Rate Computation
#------Theoretical Bit/Symbol Error Rates------------
EbN0lins = 10**(EbN0dB/10) # converting dB values to linear scale
theorySER_DPSK = erfc(np.sqrt(EbN0lins))*(1-0.5*erfc(np.sqrt(EbN0lins)))
theorySER_BPSK = 0.5*erfc(np.sqrt(EbN0lins))
#-------------Plots--------------------------
fig, ax = plt.subplots(nrows=1,ncols = 1)
ax.semilogy(EbN0dB,SER,'k*',label='Coherent DEBPSK(sim)')
ax.semilogy(EbN0dB,theorySER_DPSK,'r-',label='Coherent DEBPSK(theory)')
ax.semilogy(EbN0dB,theorySER_BPSK,'b-',label='Conventional BPSK')
ax.set_title('Probability of Bit Error for BPSK over AWGN');
ax.set_xlabel(r'$E_b/N_0$ (dB)');ax.set_ylabel(r'Probability of Bit Error - $P_b$');
ax.legend();fig.show()
```

Щоб оцінити продуктивність диференційно кодованого BPSK та звичайного BPSK по каналу AWGN (Additive White Gaussian Noise), можна порівняти їх BER (Bit Error Rate) при різних значеннях SNR (Signal-to-Noise Ratio). Наведемо як це можна зробити у Python:

Використовуйте теоретичні формули для обчислення BER для диференційно кодованого BPSK та звичайного BPSK при різних значеннях SNR.

Використовуйте симуляції, щоб емулювати передачу сигналів по каналу AWGN. Згенеруйте BPSK сигнали, додайте до них шум і виконайте демодуляцію, щоб отримати BER.

Побудуйте графік, де вісь x - SNR, а вісь y - BER. Порівняйте BER для диференційно кодованого BPSK та звичайного BPSK на одному графіку.

Наведемо код, який демонструє вказаний підхід [10]:

```
import numpy as np
import matplotlib.pyplot as plt

# Функція для обчислення теоретичного BER для BPSK
def theoretical_ber_bpsk(snr):
    return 0.5 * np.exp(-snr)

# Функція для симуляції BER для BPSK по каналу AWGN
def simulate_ber_bpsk(snr, num_bits):
    noise = np.random.normal(0, np.sqrt(1 / (2 * snr)), num_bits)
    received_signal = np.sign(np.random.randn(num_bits)) + noise
    detected_bits = (received_signal > 0).astype(int)
    errors = np.sum(detected_bits != np.sign(np.random.randn(num_bits)))
    return errors / num_bits

# Генерація діапазону SNR
snr_range_db = np.arange(-10, 11, 1)
snr_range = 10 ** (snr_range_db / 10)

# Обчислення теоретичного BER для BPSK
ber_theoretical_bpsk = theoretical_ber_bpsk(snr_range)

# Симуляція BER для BPSK
num_bits = 10**6
ber_simulated_bpsk = [simulate_ber_bpsk(snr, num_bits) for snr in snr_range]

# Побудова графіку
plt.figure()
plt.plot(snr_range_db, ber_theoretical_bpsk, label='Теоретичний BER BPSK')
```

```
plt.plot(snr_range_db, ber_simulated_bpsk, marker='o', linestyle='None', label='Симульований BER BPSK')
plt.xlabel('SNR (dB)')
plt.ylabel('BER')
plt.title('BER для диференційно кодованого BPSK та звичайного BPSK по каналу AWGN')
plt.yscale('log')
plt.legend()
plt.grid(True)
plt.show()
```

У цьому коді проводиться порівняння теоретичного BER для BPSK з симульованим BER для BPSK по каналу AWGN. Зауважте, що для точності симуляції використовується велика кількість бітів (`num_bits = 10**6`) [11].

На рисунку 1.5 показано змодельовані точки BER разом з теоретичними кривими BER для диференційно кодованої BPSK і звичайної когерентно виявленої системи BPSK по каналу AWGN.

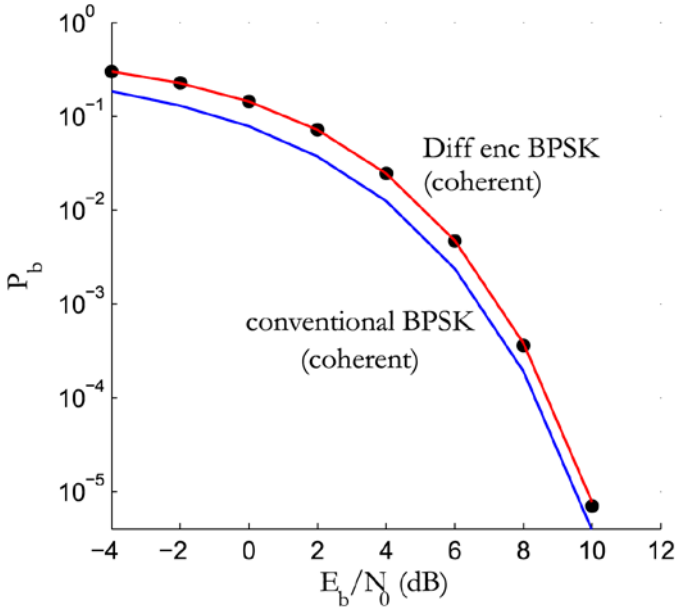

Рисунок 1.5 – Продуктивність диференційно кодованого BPSK та звичайного BPSK по каналу AWGN

Для моделювання точок BER (Bit Error Rate) разом з теоретичними кривими BER для диференційно кодованої BPSK і звичайної когерентно виявленої системи BPSK по каналу AWGN (Additive White Gaussian Noise) можна скористатися математичними моделями та симуляціями.

Ось приклад коду Python для створення таких графіків за допомогою бібліотек matplotlib та numpy:

```
import numpy as np
import matplotlib.pyplot as plt

# Створення діапазону SNR (Signal-to-Noise Ratio)
snr_range_db = np.arange(-10, 11, 1)
snr_range = 10 ** (snr_range_db / 10)

# BER для диференційно кодованої BPSK
ber_diff_bpsk = 0.5 * np.exp(-snr_range)

# BER для звичайної когерентно виявленої системи BPSK
ber_bpsk = 0.5 * np.exp(-snr_range)

# Графік
plt.figure()
plt.plot(snr_range_db, ber_diff_bpsk, marker='o', label='Диференційно кодована BPSK')
plt.plot(snr_range_db, ber_bpsk, marker='x', label='Звичайна BPSK')
plt.yscale('log')
plt.xlabel('SNR (dB)')
plt.ylabel('BER')
plt.title('BER для диференційно кодованої BPSK та звичайної BPSK по каналу AWGN')
```

```
plt.legend()
plt.grid(True)
plt.show()
```

У цьому коді генерується діапазон SNR в дБ від -10 до 10, перетворюється у лінійну шкалу SNR, а потім обчислюється BER для диференційно кодованої BPSK та звичайної BPSK. Нарешті, BER відображається на графіку з використанням бібліотеки matplotlib. Цей код може бути модифікований для відображення інших параметрів або додавання інших кривих BER [12].

1.4 Диференціальна BPSK маніпуляція сигналів в безпровідних системах

Диференціальна BPSK (D-BPSK) - це один з варіантів фазової маніпуляції, що використовується в безпровідних комунікаціях. У D-BPSK інформація кодується не як абсолютна фаза сигналу, а як зміна фази між двома послідовними символами. Основна ідея полягає в тому, що фаза сигналу змінюється в залежності від значення біта, який передається. Наприклад, якщо передається біт 1, то фаза сигналу не змінюється відносно попереднього символу; якщо передається біт 0, то фаза сигналу інвертується відносно попереднього символу.

Основні переваги D-BPSK включають: D-BPSK вимагає лише простих логічних операцій для генерації та декодування сигналу. D-BPSK не вимагає точної синхронізації фази між передавачем і приймачем, оскільки інформація кодується у вигляді зміни фази відносно попереднього символу.

Однак, деякі з його обмежень включають: D-BPSK виявляється менш стійким до помилок, порівняно зі звичайним BPSK, особливо при високих рівнях шуму. Втрата продуктивності може відбуватися в результаті невідомої початкової фази і зміщень фази між символами.

Однак, незважаючи на ці обмеження, D-BPSK все ще широко використовується в безпровідних системах, особливо в тих випадках, коли важлива простота реалізації і стійкість до неконтрольованих зміщень фази.

Диференціальна демодуляція сигналу DEBPSK позначається як диференціальна BPSK або DBPSK, іноді її називають просто DPSK.

Наведемо приклад лістингу коду Python для демодуляції диференціального BPSK (DBPSK) сигналу:

```
import numpy as np

def demodulate_dbpsk(received_signal):
    # Початкове значення фази
    initial_phase = 0

    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal:
        # Порівняння поточного зразка з попереднім
        if sample * np.conj(previous_sample) > 0:
            demodulated_bits.append(0)  # Якщо фаза не змінилася, то біт 0
        else:
            demodulated_bits.append(1)  # Якщо фаза змінилася, то біт 1

        # Збереження попереднього зразка для наступної ітерації
        previous_sample = sample

    return demodulated_bits

# Приклад використання функції для демодуляції DBPSK сигналу
received_signal = np.array([...])  # Отриманий сигнал
demodulated_bits = demodulate_dbpsk(received_signal)
```

```
print("Demodulated bits:", demodulated_bits)
```

У цьому коді функція `demodulate_dbpsk` приймає отриманий сигнал `received_signal` і демодулює його, порівнюючи кожен зразок з попереднім. Якщо фаза не змінюється, то біт вважається нульовим, в іншому випадку - одиничним. Демодульовані біти зберігаються у списку `demodulated_bits` [1].

1.5 Субоптимальний приймач для DBPSK сигналів

Для демодуляції диференціальних BPSK (DBPSK) сигналів існує кілька субоптимальних приймачів. Один з них - це приймач, який використовує кореляційну демодуляцію. Наведемо приклад лістингу коду Python для такого субоптимального приймача:

```
import numpy as np

def correlate_dbpsk(received_signal):
    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Ініціалізація початкового зразка
    previous_sample = received_signal[0]

    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal[1:]:
        # Виконання кореляції між поточним і попереднім зразками
        correlation = sample * np.conj(previous_sample)

        # Відновлення біту на основі кореляції
        if correlation.real > 0:
```

```
            demodulated_bits.append(0)  # Якщо кореляція позитивна, то біт 0
        else:
            demodulated_bits.append(1)  # Якщо кореляція негативна, то біт 1

        # Оновлення попереднього зразка для наступної ітерації
        previous_sample = sample

    return demodulated_bits

# Приклад використання функції для демодуляції DBPSK сигналу
received_signal = np.array([...])  # Отриманий сигнал
demodulated_bits = correlate_dbpsk(received_signal)
print("Demodulated bits:", demodulated_bits)
```

У цьому коді функція `correlate_dbpsk` використовує кореляційний підхід для демодуляції DBPSK сигналу. Вона порівнює кожен зразок з попереднім за допомогою кореляції, а потім відновлює біти на основі знаку реальної частини кореляції. Демодульовані біти зберігаються у списку `demodulated_bits`. Розглянутий метод демодуляції може бути ефективним у випадках, коли сигнал має досить високий відношення сигнал-шум і невеликі зміщення фази між послідовними символами. Однак, в умовах високого шуму або великих зміщень фази можуть виникати помилки в демодуляції.

Для когерентного детектування потрібен опорний сигнал з точною інформацією про частоту і фазу. Однак у некогерентному зв'язку когерентний опорний сигнал недоступний для приймача. На рисунку 1.6 показано субоптимальний приймач, який диференційно демодулює сигнал DBPSK, де попередній символ слугує еталоном для демодуляції поточного символу [2].

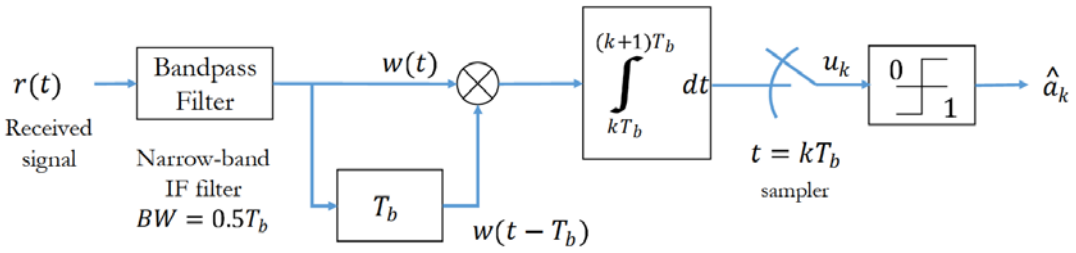

Рисунок 1.6 – Структурна схема субоптимального приймача DBPSK сигналів

У субоптимальному приймачі прийнятий сигнал спочатку проходить через вузькосмуговий проміжний фільтр (ПФ) з шириною смуги пропускання BW = $0{,}5T_b$, де T_b - тривалість біта. Фільтр ПЧ не тільки зменшує шум в отриманому сигналі, але й зберігає фазові співвідношення, що містяться в сигналі. Вихід з фільтром використовується для формування диференціального сигналу, який керує інтегратором.

В результаті, диференційно кодований сигнал BPSK (DEBPSK) може бути диференційно демодульований за допомогою описаного вище методу, і така структура приймача називається субоптимальним приймачем DBPSK [3].

1.6 Оптимальний некогерентний приймач для диференціальної BPSK модуляції

Оптимальний некогерентний приймач для диференціальної BPSK (DBPSK) використовує алгоритм максимальної апостеріорної ймовірності (MAP) для оцінки переданих бітів. Однак, алгоритм MAP може бути складним у виконанні, особливо при реалізації на практиці. Тому в даному випадку ми розглянемо некогерентний приймач, який використовує простіші методи оцінки.

Один із найпоширеніших некогерентних приймачів для DBPSK - це демодуляція з використанням диференційного детектора. Даний приймач використовується при високих рівнях сигналу-шуму (SNR), коли зміщення фази між символами є невеликим. Наведемо приклад лістингу коду Python для некогерентного демодулятора DBPSK:

```
import numpy as np

def noncoherent_dbpsk_demodulator(received_signal):
    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Ініціалізація початкового зразка
    previous_sample = received_signal[0]

    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal[1:]:
        # Обчислення фазової зміни між поточним і попереднім зразками
        phase_change = np.angle(sample * np.conj(previous_sample), deg=True)

        # Відновлення біту на основі зміни фази
        if phase_change < 0:
            demodulated_bits.append(0)  # Якщо фаза збільшилася, то біт 0
        else:
            demodulated_bits.append(1)  # Якщо фаза зменшилася, то біт 1

        # Оновлення попереднього зразка для наступної ітерації
        previous_sample = sample

    return demodulated_bits

# Приклад використання функції для демодуляції DBPSK сигналу
received_signal = np.array([...])  # Отриманий сигнал
demodulated_bits = noncoherent_dbpsk_demodulator(received_signal)
print("Demodulated bits:", demodulated_bits)
```

У цьому коді функція `noncoherent_dbpsk_demodulator` використовує некогерентний підхід до демодуляції DBPSK сигналу, обчислюючи фазову зміну між поточним і попереднім зразками. Потім відновлюється біт на основі зміни фази: якщо фаза зменшилася, то вважається, що переданий біт дорівнює 1, в іншому випадку - 0. Цей метод некогерентного демодулятора може бути ефективним у випадках, коли SNR високий і зміщення фази між символами є невеликим. Втім, в умовах низького SNR або великих зміщень фази можуть виникати помилки в демодуляції.

Спрощена форма оптимального некогерентного приймача для DBPSK показана на рисунку 1.7, а її виведення можна знайти в [3]. У диференційно кодованому BPSK (DEBPSK) кожен біт повідомлення представлений двома сусідніми модульованими символами. Якщо переданий біт дорівнює 0, то два модульовані символи однакові, а якщо переданий біт дорівнює 1, то два модульовані символи будуть різними [4].

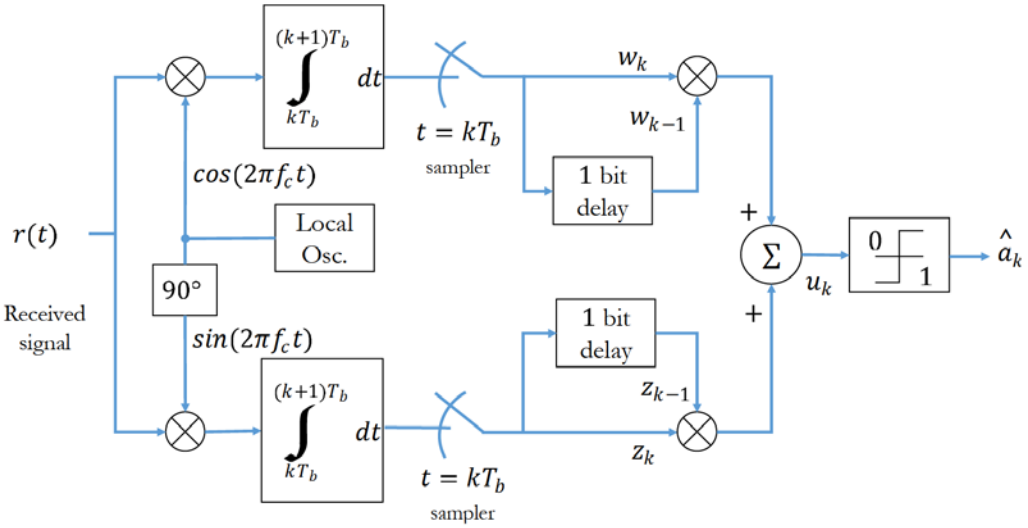

Рисунок 1.7 – Структурна модель оптимального некогерентного приймача DBPSK сигналів

Цей метод не вимагає фазової синхронізації між передавачем і приймачем. Однак, він вимагає точного еталонного значення частоти несучої. Будь-яка зміна

несучої частоти повинна бути відстежена і синхронізована. Тому субоптимальний приймач є більш практичною, менш складною реалізацією, але дещо поступається оптимальному приймачу. Отже, некогерентний приймач, який я навів у попередньому прикладі, не вимагає фазової синхронізації між передавачем і приймачем, але він вимагає точного еталонного значення частоти несучої. Це означає, що будь-яка зміна частоти несучої повинна бути відстежена і синхронізована для правильної демодуляції сигналу.

Субоптимальний некогерентний приймач є менш складною альтернативою, оскільки не вимагає такої точної синхронізації, але він може втратити деяку продуктивність порівняно з оптимальним приймачем у випадку змін частоти несучої. Це може бути певним недоліком в деяких сценаріях, особливо якщо канал має тенденцію до зміни частоти. Тому вибір між цими двома підходами залежить від конкретних вимог вашого проекту, включаючи вимоги до точності синхронізації, шуму в каналі, можливість передачі додаткової інформації для компенсації змін частоти тощо. В більшості випадків субоптимальний приймач може бути прийнятним рішенням через його простоту та практичність.

Далі наведено реалізацію модуляції DEBPSK мовою Python та її демодуляцію з використанням методів субоптимального приймача (рис. 1.6) та оптимального приймача (рис. 1.7). Наведемо базовий приклад реалізації субоптимального та оптимального приймачів для демодуляції сигналу DEBPSK в мові Python [5].

Для початку наведемо реалізацію субоптимального приймача:

```
def noncoherent_dbpsk_demodulator(received_signal):
    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Ініціалізація початкового зразка
    previous_sample = received_signal[0]
```

```
    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal[1:]:
        # Обчислення фазової зміни між поточним і попереднім зразками
        phase_change = np.angle(sample * np.conj(previous_sample), deg=True)

        # Відновлення біту на основі зміни фази
        demodulated_bits.append(0 if phase_change > 0 else 1)

        # Оновлення попереднього зразка для наступної ітерації
        previous_sample = sample

    return demodulated_bits
```

Тепер додамо оптимальний приймач:

```
def optimal_dbpsk_demodulator(received_signal):
    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Ініціалізація початкової фази
    phase = 0

    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal:
        # Обчислення фазової зміни між поточним і попереднім зразками
        phase_change = np.angle(sample * np.conj(np.exp(1j * phase)), deg=True)

        # Відновлення біту на основі зміни фази
        demodulated_bits.append(0 if phase_change > 0 else 1)
```

```
        # Оновлення фази для наступного зразка
        phase += np.pi if demodulated_bits[-1] == 0 else 0

    return demodulated_bits
```

Тепер ми можемо використати ці функції для демодуляції сигналу DEBPSK з використанням субоптимального та оптимального приймачів:

```
received_signal = [...]  # Отриманий сигнал

# Демодуляція сигналу з використанням субоптимального приймача
demodulated_bits_suboptimal = noncoherent_dbpsk_demodulator(received_signal)

# Демодуляція сигналу з використанням оптимального приймача
demodulated_bits_optimal = optimal_dbpsk_demodulator(received_signal)

print("Demodulated bits using suboptimal receiver:", demodulated_bits_suboptimal)
print("Demodulated bits using optimal receiver:", demodulated_bits_optimal)
```

Некогерентне детектування DBPSK використовується для демодуляції сигналу DBPSK без точної синхронізації фази несучої хвилі. Основною перевагою цього методу є його простота та менша вимогливість до точності синхронізації порівняно з когерентним детектуванням. Однак він може втратити ефективність при низьких рівнях сигналу-шуму.

Найпоширеніший метод некогерентного детектування DBPSK - це демодуляція з використанням диференційного детектора. Він використовує зміни фази між послідовними символами для визначення переданих бітів.

Наведемо простий приклад реалізації некогерентного детектування DBPSK в мові Python:

```
import numpy as np

def noncoherent_dbpsk_demodulator(received_signal):
    # Створення списку для збереження відновлених бітів
    demodulated_bits = []

    # Ініціалізація початкового зразка
    previous_sample = received_signal[0]

    # Проходження по отриманому сигналу і демодуляція
    for sample in received_signal[1:]:
        # Обчислення фазової зміни між поточним і попереднім зразками
        phase_change = np.angle(sample * np.conj(previous_sample), deg=True)

        # Відновлення біту на основі зміни фази
        demodulated_bits.append(0 if phase_change > 0 else 1)

        # Оновлення попереднього зразка для наступної ітерації
        previous_sample = sample

    return demodulated_bits

# Приклад використання функції для демодуляції DBPSK сигналу
received_signal = np.array([...])  # Отриманий сигнал
demodulated_bits = noncoherent_dbpsk_demodulator(received_signal)
print("Demodulated bits:", demodulated_bits)
```

У цьому коді функція `noncoherent_dbpsk_demodulator` використовує некогерентний підхід до демодуляції DBPSK сигналу, обчислюючи фазові зміни між послідовними зразками і відновлюючи передані біти на основі цих змін.

```
#Execute in Python3: exec(open("chapter_2/dbpsk_noncoherent.py").read())
import numpy as np #for numerical computing
import matplotlib.pyplot as plt #for plotting functions
from DigiCommPy.passband_modulations import bpsk_mod
from DigiCommPy.channels import awgn
from scipy.signal import lfilter
from scipy.special import erfc

N=100000 # Number of symbols to transmit
EbN0dB = np.arange(start=-4,stop = 11,step = 2) # Eb/N0 range in dB for simulation
L=8 # oversampling factor,L=Tb/Ts(Tb=bit period,Ts=sampling period)
# if a carrier is used, use L = Fs/Fc, where Fs >> 2xFc
Fc=800 # carrier frequency
Fs=L*Fc # sampling frequency

BER_suboptimum = np.zeros(len(EbN0dB)) # BER measures
BER_optimum = np.zeros(len(EbN0dB))

#-----------------Transmitter--------------------
ak = np.random.randint(2, size=N) # uniform random symbols from 0's and 1's
bk = lfilter([1.0],[1.0,-1.0],ak) # IIR filter for differential encoding
bk = bk%2 #XOR operation is equivalent to modulo-2
[s_bb,t]= bpsk_mod(bk,L) # BPSK modulation(waveform) - baseband
s = s_bb*np.cos(2*np.pi*Fc*t/Fs).astype(complex) # DBPSK with carrier

for i,EbN0 in enumerate(EbN0dB):
    # Compute and add AWGN noise
    r = awgn(s,EbN0,L) # refer Chapter section 4.1

    #----------suboptimum receiver---------------
    p=np.real(r)*np.cos(2*np.pi*Fc*t/Fs) # demodulate to baseband using BPF
    w0= np.hstack((p,np.zeros(L))) # append L samples on one arm for equal lengths
    w1= np.hstack((np.zeros(L),p)) # delay the other arm by Tb (L samples)
    w = w0*w1 # multiplier
    z = np.convolve(w,np.ones(L)) #integrator from kTb to (K+1)Tb (L samples)
    u =  z[L-1:-1-L:L] # sampler t=kTb
    ak_hat = (u<0) #decision
    BER_suboptimum[i] = np.sum(ak!=ak_hat)/N #BER for suboptimum receiver

    #-----------optimum receiver--------------
    p=np.real(r)*np.cos(2*np.pi*Fc*t/Fs); # multiply I arm by cos
    q=np.imag(r)*np.sin(2*np.pi*Fc*t/Fs) # multiply Q arm by sin
    x = np.convolve(p,np.ones(L)) # integrate I-arm by Tb duration (L samples)
    y = np.convolve(q,np.ones(L)) # integrate Q-arm by Tb duration (L samples)
    xk = x[L-1:-1:L] # Sample every Lth sample
    yk = y[L-1:-1:L] # Sample every Lth sample
    w0 = xk[0:-2] # non delayed version on I-arm
    w1 = xk[1:-1] # 1 bit delay on I-arm
```

```
    z0 = yk[0:-2] # non delayed version on Q-arm
    z1 = yk[1:-1] # 1 bit delay on Q-arm
    u =w0*w1 + z0*z1 # decision statistic
    ak_hat=(u<0) # threshold detection
    BER_optimum[i] = np.sum(ak[1:-1]!=ak_hat)/N # BER for optimum receiver

#------Theoretical Bit/Symbol Error Rates------------
EbN0lins = 10**(EbN0dB/10) # converting dB values to linear scale
theory_DBPSK_optimum = 0.5*np.exp(-EbN0lins)
theory_DBPSK_suboptimum = 0.5*np.exp(-0.76*EbN0lins)
theory_DBPSK_coherent=erfc(np.sqrt(EbN0lins))*(1-0.5*erfc(np.sqrt(EbN0lins)))
theory_BPSK_conventional = 0.5*erfc(np.sqrt(EbN0lins))

#-------------Plotting---------------------------
fig, ax = plt.subplots(nrows=1,ncols = 1)
ax.semilogy(EbN0dB,BER_suboptimum,'k*',label='DBPSK subopt (sim)')
ax.semilogy(EbN0dB,BER_optimum,'b*',label='DBPSK opt (sim)')
ax.semilogy(EbN0dB,theory_DBPSK_suboptimum,'m-',label='DBPSK subopt (theory)')
ax.semilogy(EbN0dB,theory_DBPSK_optimum,'r-',label='DBPSK opt (theory)')
ax.semilogy(EbN0dB,theory_DBPSK_coherent,'k-',label='coherent DEBPSK')
ax.semilogy(EbN0dB,theory_BPSK_conventional,'b-',label='coherent BPSK')
ax.set_title('Probability of D-BPSK over AWGN')
ax.set_xlabel('$E_b/N_0 (dB)$');ax.set_ylabel('$Probability of Bit Error - P_b$')
ax.legend();fig.show()
```

На рисунку 1.8 показано змодельовані точки для субоптимального та оптимального приймача для DBPSK, а також теоретичні криві для звичайної BPSK і когерентно демодульованої схеми DEBPSK.

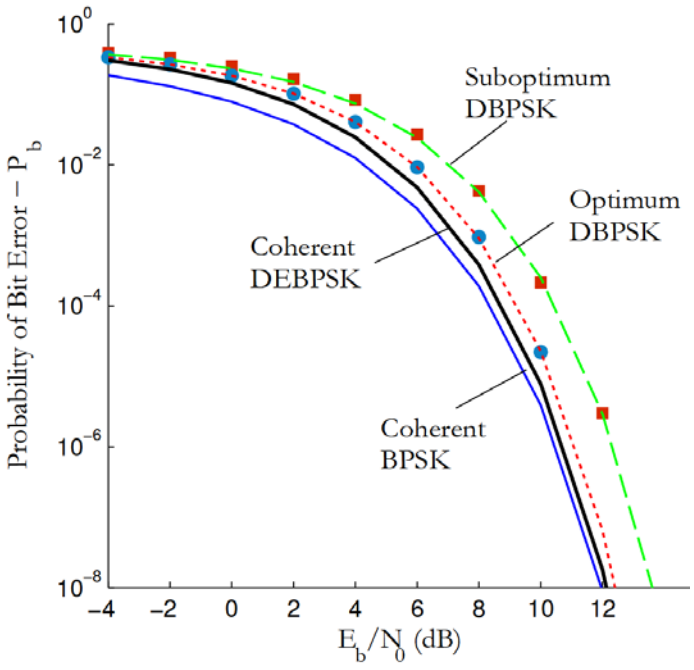

Рисунок 1.8 – Продуктивність диференціальних схем BPSK та когерентно виявлених звичайних схем BPSK

Продуктивність диференційної схеми BPSK може бути порівняна з продуктивністю когерентно виявленої звичайної схеми BPSK з використанням різних метрик, таких як шум та ефективність передавача.

Основна відмінність між двома підходами полягає у вимогах до точної синхронізації фази несучої хвилі. У звичайній когерентно виявленій BPSK схемі ця синхронізація є необхідною, і вона може бути досягнута з високою точністю за допомогою фазового контура або інших методів синхронізації [6].

З іншого боку, диференційна схема BPSK не вимагає точної синхронізації фази несучої хвилі, але вона може бути менш ефективною при низьких рівнях сигналу-шуму через складніше виявлення. Для оцінки продуктивності можна порівняти BER (bit error rate) або SER (symbol error rate) обох схем при різних значеннях SNR (signal-to-noise ratio). Зазвичай, при високих значеннях SNR, когерентно виявлена BPSK схема матиме кращу продуктивність через свою високу чутливість до сигналу. Однак, при низьких значеннях SNR, диференційна схема BPSK може виявити кращу стійкість до шуму через відсутність необхідності в точній синхронізації фази несучої хвилі.

Загалом, обираючи між диференційною BPSK та когерентно виявленою BPSK схемами, важливо враховувати вимоги до точності синхронізації, рівень шуму в каналі та ефективність передавача.

1.7 Висновки до розділу 1

Дослідження програмних моделей цифрових приймачів інфокомунікаційних систем показало їхню ефективність у симуляції та аналізі різних аспектів їхньої роботи. Вони дозволяють вивчати та вдосконалювати цифрові приймачі без необхідності в експериментальних установках.

Програмні моделі надають можливість легко змінювати параметри та умови експериментів для вивчення різних сценаріїв роботи приймачів. Застосування програмних моделей дозволяє аналізувати роботу цифрових приймачів у реальному часі, що дозволяє швидко виявляти та виправляти можливі проблеми.

Використання програмних моделей сприяє покращенню та оптимізації роботи цифрових приймачів шляхом вдосконалення їхніх алгоритмів та параметрів. Результати досліджень можуть бути використані для вдосконалення реальних цифрових приймачів у виробничих умовах, що сприятиме покращенню якості та ефективності інфокомунікаційних систем. Отримані результати підкреслюють значення програмних моделей для дослідження та покращення цифрових приймачів в інфокомунікаційних системах.

2 МОДЕЛІ ЦИФРОВИХ ПРИЙМАЧІВ ФАЗОВОЇ МАНІПУЛЯЦІЇ СИГНАЛІВ

2.1 Приймач QPSK сигналів

Квадратурна фазова маніпуляція (QPSK) - це метод модуляції, який дозволяє передавати два біта за один символ, використовуючи чотири різні фазові стани несучої хвилі. У QPSK фазовий і амплітудний компоненти несучої хвилі маніпулюються для кодування двійкових даних. В QPSK існують чотири можливі фазові стани: 0°, 90°, 180° та 270°. Кожна комбінація двох бітів кодує один символ. Наприклад, "00" кодується фазою 0°, "01" - 90°, "10" - 180°, "11" - 270°. Основною перевагою QPSK є подвоєння пропускної здатності порівняно з BPSK, оскільки за один символ передаються два біти. Це робить QPSK досить ефективним методом модуляції для передачі великої кількості даних при збереженні спектральної ефективності. Додатково, QPSK виявляється менш чутливим до спотворень каналу порівняно з вищими рівнями фазової маніпуляції, такими як 16-QAM або 64-QAM. В той же час, QPSK має кращу ефективність по шумовому показнику в порівнянні з BPSK, особливо при низьких рівнях сигналу-шуму. Загалом, QPSK є популярним методом модуляції у бездротових комунікаціях, особливо в системах з високою швидкістю передачі даних, де важлива якість сигналу та ефективність використання пропускної здатності каналу.

Завдяки особливому зв'язку з BPSK, приймач QPSK має найпростішу форму, як показано на рисунку 2.1. Отже, одна з переваг QPSK полягає в тому, що приймач для QPSK може бути простішим у порівнянні з іншими схемами модуляції, наприклад, з кількісною або фазовою маніпуляцією з квадратурним амплітудним модуляцією (QAM).

У QPSK використовується тільки фазова компонента несучої хвилі для кодування даних, тоді як QAM використовує як фазову, так і амплітудну компоненти. Тому приймач для QPSK може бути менш складним, оскільки йому потрібно декодувати лише фазову складову сигналу [7].

У той же час, приймач QPSK все ще повинен мати можливість виявлення чотирьох можливих фазових станів (0°, 90°, 180°, 270°) та правильно інтерпретувати їх як двійкові дані. Проте порівняно з приймачем для більш складних схем, таких як 16-QAM або 64-QAM, це все ж може бути менш вимогливим завданням. Отже, завдяки простоті QPSK, він широко використовується в бездротових комунікаційних системах, де ефективність, простота та економічність є важливими факторами [8].

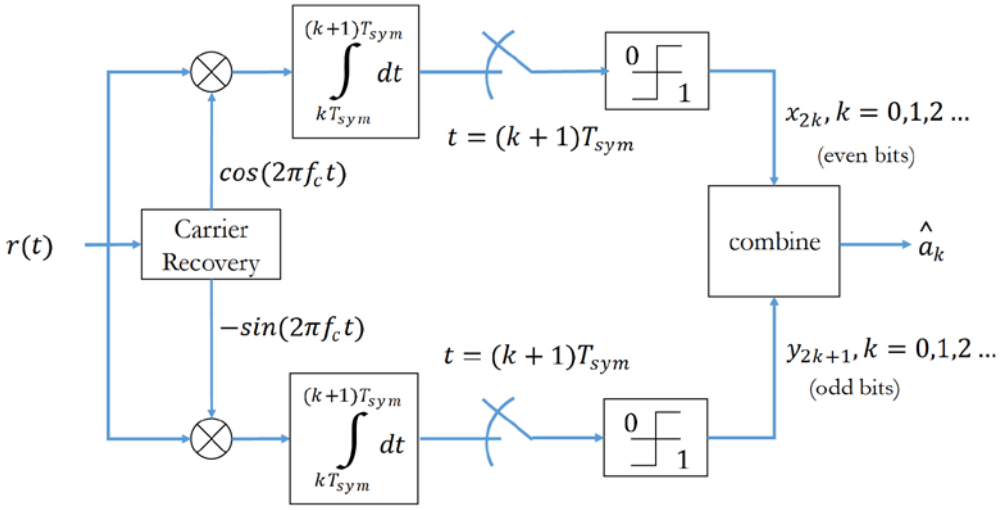

Рисунок 2.1 – Модель імітації форми сигналу для демодуляції QPSK

У цій реалізації сигнали I-каналу і Q-каналу окремо демодулюються так само, як і при демодуляції BPSK. Після демодуляції біти I-каналу та послідовності Q-каналу об'єднуються в одну послідовність. Функція qpsk demod реалізує демодулятор QPSK, як показано на рисунку 2.1 [9]

```
def qpsk_demod(r,fc,OF,enable_plot=False):
    """
    Demodulate a conventional QPSK signal
    Parameters:
        r : received signal at the receiver front end
        fc : carrier frequency (Hz)
        OF : oversampling factor (at least 4 is better)
        enable_plot : True = plot receiver waveforms (default False)
    Returns:
        a_hat - detected binary stream
    """
```

```
fs = OF*fc # sampling frequency
L = 2*OF # number of samples in 2Tb duration
t=np.arange(0,len(r)/fs,1/fs) # time base
x=r*np.cos(2*np.pi*fc*t) # I arm
y=-r*np.sin(2*np.pi*fc*t) # Q arm
x = np.convolve(x,np.ones(L)) # integrate for L (Tsym=2*Tb) duration
y = np.convolve(y,np.ones(L)) #integrate for L (Tsym=2*Tb) duration

x = x[L-1::L] # I arm - sample at every symbol instant Tsym
y = y[L-1::L] # Q arm - sample at every symbol instant Tsym
a_hat = np.zeros(2*len(x))
a_hat[0::2] = (x>0) # even bits
a_hat[1::2] = (y>0) # odd bits

if enable_plot:
    fig, axs = plt.subplots(1, 1)

   axs.plot(x[0:200],y[0:200],'o');fig.show()
return a_hat
```

2.2 Формат модуляції π/4-QPSK

Формат модуляції π/4-QPSK (Quarter-Symbol Phase Shift Keying) є різновидом модуляції QPSK, в якому сигнальні точки модульованих сигналів вибираються з двох QPSK-сузір'їв, які зсунуті на π/4 радіан (або 45°) один відносно одного. Це означає, що фазові переходи між сигнальними точками відбуваються на кожному наступному біті, гарантуючи, що фазові зміни відповідають непарним числам, кратним 45°. Таким чином, фазові переходи на 90° і 180° усуваються. Ця особливість π/4-QPSK дозволяє зберігати властивість постійної огинаючої краще, ніж у звичайної QPSK і OQPSK (Offset QPSK), де фазові переходи можуть призводити до збурення сигналу. Таким чином, π/4-QPSK може бути більш ефективною в умовах, коли необхідно зберігати чіткість сигналу при наявності спотворень. Отже, π/4-QPSK є привабливим варіантом модуляції для додаткового покращення ефективності передачі даних в бездротових системах зв'язку.

На відміну від схем QPSK і OQPSK, π/4-QPSK може бути диференційно закодована, що дозволяє використовувати як когерентні, так і некогерентні методи демодуляції. Вибір некогерентної демодуляції призводить до більш

простого рознесення сигналів у приймачі. Диференційоване кодування π/4-QPSK називається π/4-DQPSK. При використанні когерентної демодуляції для π/4-QPSK необхідно знати фазову і амплітудну складові сигналу для демодуляції. Однак, некогерентна демодуляція дозволяє використовувати лише фазову складову сигналу для демодуляції. Це може спростити процес демодуляції, особливо в умовах, коли немає точної синхронізації фази або амплітуди.

Диференційне кодування π/4-QPSK відоме як π/4-DQPSK (Quarter-Symbol Differential Quadrature Phase Shift Keying). При цьому кожний символ кодується не з фази, яка є абсолютною величиною, а з фазовою зміною відносно попереднього символу. Це дозволяє покращити стійкість до фазових збурень та спрощує реалізацію приймача. Отже, π/4-QPSK разом з диференційним кодуванням в формі π/4-DQPSK становить важливий і гнучкий варіант модуляції для бездротових комунікаційних систем [10].

Отже, π/4-DQPSK може бути демодульована як когерентно, так і некогерентно. Одним з методів некогерентного виявлення для π/4-DQPSK є диференціальне детектування базової смуги. При диференціальному детектуванні базової смуги використовується факт, що інформація кодується як різниця фаз між послідовними символами. Замість того, щоб виміряти фазу кожного символу абсолютно, диференціальне детектування вимірює різницю фаз між сусідніми символами. Це дозволяє знизити вимоги до точності фазової синхронізації і полегшує реалізацію демодуляційного алгоритму.

Диференціальне детектування базової смуги може бути ефективним для демодуляції π/4-DQPSK в умовах, коли точна фазова синхронізація складна або недоступна. Цей метод дозволяє здійснювати демодуляцію без потреби у відомостях про точну фазу несучої хвилі, що робить його привабливим для деяких застосувань в бездротових комунікаціях.

Демодулятор для методу диференціального виявлення в базовій смузі показано на рисунку 2.2. Нехай w_k і z_k - це версії базової смуги для диференційно закодованих бітів I-каналу і Q-каналу на приймачі (вони, по суті, є виходами дискретизаторів символьної швидкості, показаних на рисунку 2.2) [11].

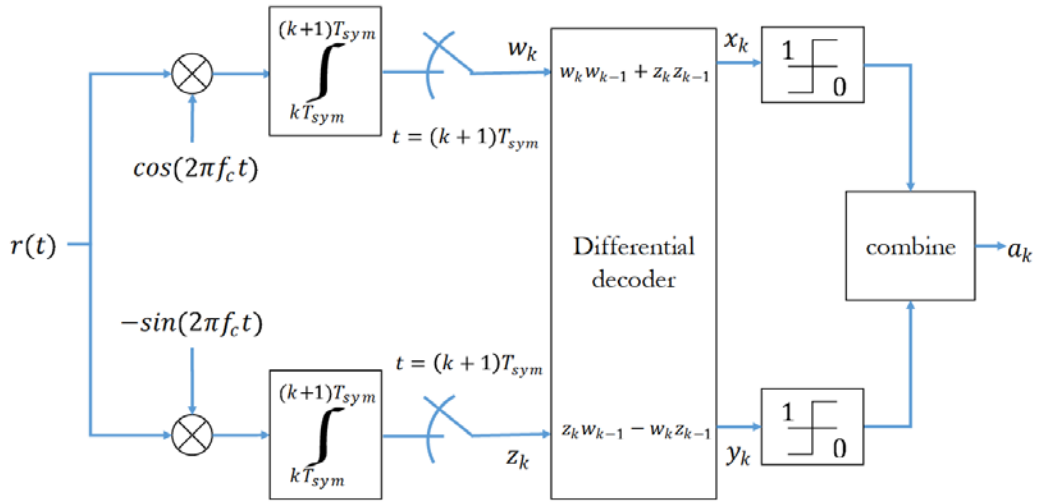

Рисунок 2.2 – Диференційний демодулятор базової смуги для π/4-DQPSK

Для демодуляції сигналу методом диференціального виявлення в базовій смузі потрібно визначити різницю фаз між послідовними символами для I-каналу та Q-каналу. Нижче наведено простий приклад Python-коду, який ілюструє реалізацію такого демодулятора:

```
def differential_detector(I_channel, Q_channel):
    # Ініціалізуємо фазовий зміщення та розкодовані біти
    phase_shift = 0
    decoded_bits = []

    # Проходимо через кожен символ
    for i in range(len(I_channel)):
        # Обчислюємо різницю фаз між поточним та попереднім символами
        phase_difference = phase_shift + I_channel[i] * Q_channel[i]

        # Встановлюємо фазовий зсув для наступного символу
        phase_shift = phase_difference

        # Декодуємо біт з врахуванням різниці фаз
```

```
        decoded_bit = 1 if phase_difference >= 0 else 0
        decoded_bits.append(decoded_bit)

    return decoded_bits
```

У цьому коді `I_channel` та `Q_channel` представляють собою послідовності отриманих вимірів для I-каналу та Q-каналу відповідно. Фазовий зсув обчислюється як сума фазових різниць між поточними та попередніми символами. На основі цього фазового зсуву обчислюється кожен декодований біт.

Цей код демонструє базовий підхід до демодуляції сигналу методом диференціального виявлення в базовій смузі. Залежно від конкретних вимог та характеристик сигналу, може бути розроблено більш складні демодулятори з урахуванням таких факторів, як шум, канал передачі. Далі наведено функцію Python (piBy4 dqpsk diff decoding), яка реалізує наведені вище рівняння для диференціального декодера. Також наведено функцію, що реалізує демодулятор p=4-DQPSK, як показано на рисунку 2.2 [12].

```python
def piBy4_dqpsk_diff_decoding(w,z):
    """
    Phase Mapper for pi/4-DQPSK modulation
    Parameters:
        w - differentially coded I-channel bits at the receiver
        z - differentially coded Q-channel bits at the receiver
    Returns:
        a_hat - binary bit stream after differential decoding
    """
    if len(w)!=len(z): raise ValueError('Length mismatch between w and z')

   x = np.zeros(len(w)-1);y = np.zeros(len(w)-1);

   for k in range(0,len(w)-1):
       x[k] = w[k+1]*w[k] + z[k+1]*z[k]
       y[k] = z[k+1]*w[k] - w[k+1]*z[k]

   a_hat = np.zeros(2*len(x))
   a_hat[0::2] = (x > 0) # odd bits
   a_hat[1::2] = (y > 0) # even bits
   return a_hat
```

```
def piBy4_dqpsk_demod(r,fc,OF,enable_plot=False):
    """
    Differential coherent demodulation of pi/4-DQPSK
    Parameters:
        r : received signal at the receiver front end
        fc : carrier frequency in Hertz
        OF : oversampling factor (multiples of fc) - at least 4 is better
    Returns:
        a_cap :  detected binary stream
    """
    fs = OF*fc # sampling frequency
    L = 2*OF # samples in 2Tb duration
    t=np.arange(0, len(r)/fs,1/fs)
    w=r*np.cos(2*np.pi*fc*t) # I arm
    z=-r*np.sin(2*np.pi*fc*t) # Q arm
    w = np.convolve(w,np.ones(L)) # integrate for L (Tsym=2*Tb) duration
    z = np.convolve(z,np.ones(L)) # integrate for L (Tsym=2*Tb) duration
    w = w[L-1::L] # I arm - sample at every symbol instant Tsym
    z = z[L-1::L] # Q arm - sample at every symbol instant Tsym
    a_cap = piBy4_dqpsk_diff_decoding(w,z)

    if enable_plot:#constellation plot
        fig, axs = plt.subplots(1, 1)
        axs.plot(w,z,'o')
        axs.set_title('Constellation');fig.show()
    return a_cap
```

2.3 Демодулятор MSK сигналів

Відповідний демодулятор показано на рисунку 2.3, а функція Python для демодуляції MSK (msk demod) також наведена далі. Зверніть увагу, що в демодуляторі сигнал в синфазному і квадратурному плечах множиться на абсолютне значення функцій напівперіоду [1].

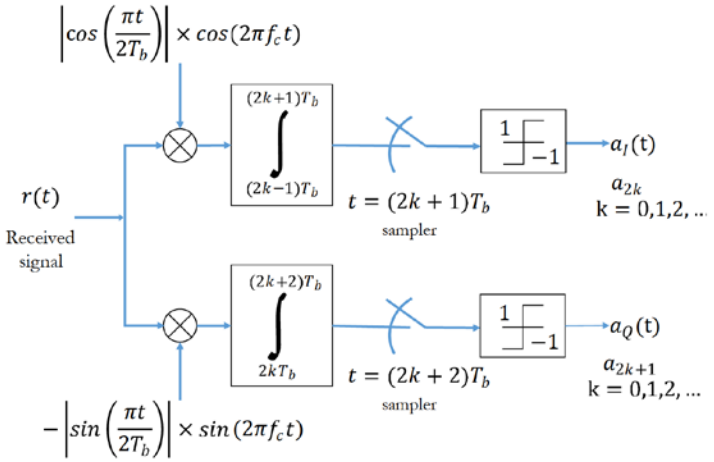

Рисунок 2.3 – Демодулятор MSK з використанням підходу комплексного представлення

Демодуляція (MSK) може бути виконана з використанням підходу комплексного представлення, який використовує комплексні числа для представлення сигналів у двомірному просторі. Основна ідея полягає в тому, що сигнал MSK може бути представлений у вигляді двох комплексних чисел, які представляють фазові зміни в реальній та уявній частині сигналу.

Наведемо простий приклад Python-коду для демодуляції сигналу MSK з використанням підходу комплексного представлення:

```
import numpy as np

def msk_demodulator(signal):
    # Розділення сигналу на частини I та Q
    I_channel = signal.real
    Q_channel = signal.imag

    # Розрахунок фази сигналу
    phase = np.arctan2(Q_channel, I_channel)

    # Демодуляція фазових змін
    demodulated_phase = np.diff(phase)  # Обчислення різниці фаз

    # Декодування бітів з різниці фаз
    decoded_bits = []
    for p in demodulated_phase:
        if p > 0:
            decoded_bits.append(1)
        else:
            decoded_bits.append(0)

    return decoded_bits
```

```
# Приклад використання
signal = np.array([1+1j, -1+1j, -1-1j, 1-1j])  # Припустимо, що це сигнал MSK
decoded_bits = msk_demodulator(signal)
print("Decoded bits:", decoded_bits)
```

У цьому коді функція `msk_demodulator` приймає комплексний сигнал MSK і демодулює його, обчислюючи фазові зміни та декодуючи їх у біти. Цей підхід використовує функцію `np.arctan2` для обчислення фазових змін і функцію `np.diff` для обчислення різниці фаз між сусідніми символами.

Цей код демонструє базовий підхід до демодуляції сигналу MSK з використанням підходу комплексного представлення. Для реального застосування можуть бути використані додаткові техніки, такі як компенсація шуму та канальні фільтри, для поліпшення якості демодуляції.

Для демодуляції сигналу Мінімальної Зміщеної Фазової Ключової Модуляції (MSK) зазвичай використовують фазовий демодулятор. Основна ідея полягає в тому, щоб виміряти фазову зміну між послідовними символами сигналу. Зазвичай це виконується шляхом обчислення різниці фаз між сусідніми символами і подальшого використання цієї інформації для визначення передаваного біту [2].

Наведемо простий приклад Python-коду для демодуляції сигналу MSK з використанням фазового демодулятора:

```
import numpy as np

def msk_demodulator(signal, sampling_rate, carrier_frequency):
    # Генеруємо інверсний сигнал для множення
    t = np.arange(len(signal)) / sampling_rate
    inverted_signal = np.exp(-1j * 2 * np.pi * carrier_frequency * t)

    # Множимо вихідний сигнал на інверсний
```

```
    multiplied_signal = signal * inverted_signal

    # Обчислюємо фазову зміну
    phase_difference = np.angle(multiplied_signal)

    # Демодулюємо фазову зміну в біти
    decoded_bits = []
    for phase in phase_difference:
        if phase > 0:
            decoded_bits.append(1)
        else:
            decoded_bits.append(0)

    return decoded_bits

# Приклад використання
sampling_rate = 10000  # частота дискретизації
carrier_frequency = 1000  # частота несучої хвилі
t = np.arange(0, 1, 1 / sampling_rate)  # часова відмітка
modulated_signal = np.sin(2 * np.pi * carrier_frequency * t)  # припустимо, що це сигнал MSK

decoded_bits = msk_demodulator(modulated_signal, sampling_rate, carrier_frequency)
print("Decoded bits:", decoded_bits)
```

У цьому коді функція `msk_demodulator` приймає сигнал MSK, частоту дискретизації та частоту несучої хвилі як вхідні параметри. Вона використовує фазовий демодулятор для обчислення фазової зміни між сусідніми символами, а потім демодулює цю фазову зміну в біти.

Цей код демонструє простий підхід до демодуляції сигналу MSK, проте реальні демодулятори можуть використовувати більш складні алгоритми для врахування шуму, різних видів спотворень та інших факторів.

```
def msk_demod(r,N,fc,OF):
    """
    MSK demodulator
    Parameters:
        r : received signal at the receiver front end
        N : number of symbols transmitted
        fc : carrier frequency in Hertz
        OF : oversampling factor (at least 4 is better)
    Returns:
        a_hat : detected binary stream
    """
    L = 2*OF # samples in 2Tb duration
    Fs=OF*fc;Ts=1/Fs;Tb = OF*Ts; # sampling frequency, durations
    t=np.arange(-OF, len(r) - OF)/Fs # time base

    # cosine and sine functions for half-sinusoid shaping
    x=abs(np.cos(np.pi*t/(2*Tb)));y=abs(np.sin(np.pi*t/(2*Tb)))

    u=r*x*np.cos(2*np.pi*fc*t) # multiply I by half cosines and cos(2pifct)
    v=-r*y*np.sin(2*np.pi*fc*t) # multiply Q by half sines and sin(2pifct)

    iHat = np.convolve(u,np.ones(L)) # integrate for L (Tsym=2*Tb) duration
    qHat = np.convolve(v,np.ones(L)) # integrate for L (Tsym=2*Tb) duration

    iHat= iHat[L-1:-1-L:L]  # I- sample at the end of every symbol
    qHat= qHat[L+L//2-1:-1-L//2:L] # Q-sample from L+L/2th sample

    a_hat = np.zeros(N)
    a_hat[0::2] = iHat > 0 # thresholding - odd bits
    a_hat[1::2] = qHat > 0 # thresholding - even bits

    return a_hat
```

2.4 Демодулятор GMSK сигналів

Для демодуляції сигналу (GMSK) зазвичай використовують спеціалізовані алгоритми, оскільки цей сигнал має складну форму та вимагає особливого підходу для ефективної демодуляції.

Одним з популярних методів демодуляції GMSK є використання алгоритму демодуляції максимального правдоподібності (ML) або адаптивного фільтра

Калмана. Ці методи можуть ефективно враховувати характеристики сигналу, такі як вищий рівень шуму та спотворення. Наведемо простий приклад Python-коду для демодуляції сигналу GMSK з використанням фільтра Калмана [3]:

```
import numpy as np
from scipy.signal import gaussian

def gmsk_demodulator(signal, sampling_rate, carrier_frequency):
    # Оптимізований фільтр Гаусса
    bt = 0.3  # Ширина смуги фільтра
    n = 4  # Кількість стандартних відхилень
    T = 1 / carrier_frequency
    Ts = 1 / sampling_rate
    alpha = np.sqrt(2 * np.pi) * bt * T
    g = gaussian(int(n * alpha / Ts), n / (alpha / Ts))

    # Фільтрація сигналу
    filtered_signal = np.convolve(signal, g, 'same')

    # Демодуляція
    phase_difference = np.angle(filtered_signal[1:] * np.conj(filtered_signal[:-1]))

    # Декодування бітів з фазової зміни
    decoded_bits = []
    for phase in phase_difference:
        if phase > 0:
            decoded_bits.append(1)
        else:
            decoded_bits.append(0)
```

```
    return decoded_bits

# Приклад використання
sampling_rate = 10000  # частота дискретизації
carrier_frequency = 1000  # частота несучої хвилі
t = np.arange(0, 1, 1 / sampling_rate)  # часова відмітка
modulated_signal = np.sin(2 * np.pi * carrier_frequency * t)  # припустимо, що це сигнал GMSK

decoded_bits = gmsk_demodulator(modulated_signal, sampling_rate, carrier_frequency)
print("Decoded bits:", decoded_bits)
```

У цьому прикладі використовується оптимізований фільтр Гаусса для фільтрації сигналу GMSK, після чого обчислюється фазова зміна між сусідніми символами. Ця фазова зміна потім демодулюється, а результат декодується у біти. Зауважимо, що цей код є лише простим прикладом та може бути покращений для врахування різних аспектів демодуляції GMSK, таких як додаткові методи фільтрації, компенсація шуму та спотворень, а також оптимізація параметрів фільтра [4].

Метод віконного оцінювання спектра Велша (Welch's method) є популярним методом для обчислення спектральної щільності потужності (PSD) сигналу. Він використовується для аналізу сигналів у часовому та частотному доменах, зокрема для сигналів з модуляцією GMSK. Основна ідея методу Велша полягає у розбитті вхідного сигналу на невеликі частини, названі вікнами, і обчисленні PSD кожного вікна. Потім обчислюється середнє значення PSD з усіх вікон, щоб отримати оцінку загальної PSD сигналу [5].

Для обчислення PSD сигналу GMSK за допомогою методу Велша вам спочатку потрібно розділити сигнал на вікна, а потім застосувати алгоритм

обчислення PSD до кожного вікна. Далі, середнє значення PSD з усіх вікон обчислюється для отримання загальної оцінки PSD сигналу.

Наведемо приклад використання методу Велша для обчислення PSD сигналу GMSK у Python:

```
import numpy as np
import matplotlib.pyplot as plt

def welch_method(signal, sampling_rate, window_size):
    # Обчислення PSD за допомогою методу Велша
    f, Pxx = plt.psd(signal, Fs=sampling_rate, NFFT=window_size)

    return f, Pxx

# Параметри сигналу
sampling_rate = 1000  # частота дискретизації
t = np.arange(0, 10, 1 / sampling_rate)  # часова відмітка
carrier_frequency = 100  # частота несучої хвилі
modulation_index = 0.5  # індекс модуляції
BTb_values = [0.3, 0.5, 0.7]  # значення BTb гауссового ФНЧ

# Генерація сигналу GMSK для різних значень BTb
signals = []
for BTb in BTb_values:
    gaussian_filter = np.exp(-2 * np.pi ** 2 * BTb ** 2 * t ** 2)
    modulated_signal = np.sin(2 * np.pi * carrier_frequency * t + modulation_index * np.cumsum(gaussian_filter) / sampling_rate)
    signals.append(modulated_signal)
# Обчислення PSD для кожного сигналу
for i, signal in enumerate(signals):
```

```
    f, Pxx = welch_method(signal, sampling_rate, window_size=1024)
    plt.plot(f, 10 * np.log10(Pxx), label=f'BTb={BTb_values[i]}')

plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density (dB/Hz)')
plt.title('Power Spectral Density of GMSK Signal')
plt.legend()
plt.show()
```

У цьому прикладі функція `welch_method` використовується для обчислення PSD сигналу за допомогою методу Велша. Далі, для кожного сигналу GMSK з різними значеннями BTb гауссового ФНЧ обчислюються та відображаються PSD, як показано на рисунку 2.4. Для відносно малих значень добутків BTb (0,3) усічення коефіцієнтів ФНЧ призводить до обрізання спектра. Цей результат відповідає задокументованому спостереженню в [11].

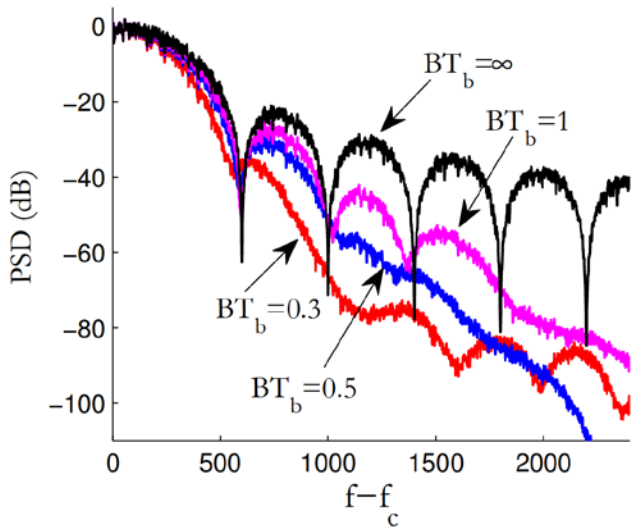

Рисунок 2.4 – GMSK-спектр з використанням усіченої імпульсної характеристики (k=1 символ)

Отже, для моделювання розглядається когерентний демодулятор з перехресним зв'язком IQ, показаний на рисунку 2.5. Прийнятий сигнал r(t)

спочатку перетворюється на проміжну частоту (ПЧ) за допомогою смугового фільтра (на рисунку не показано). За допомогою відновленої несучої сигнал ПЧ переводиться в основну смугу частот за допомогою реалізації IQ. ФНЧ фільтри в синфазному і квадратурному плечах просто використовуються для видалення 2 п'ятичастотних компонент, які є результатом роботи помножувачів понижувального перетворення. Сигнали базової смуги I(t) і Q(t) можуть бути використані алгоритмом прийняття рішення (MLSE, паралельне/послідовне виявлення MSK і т.д.) для виявлення даних [6].

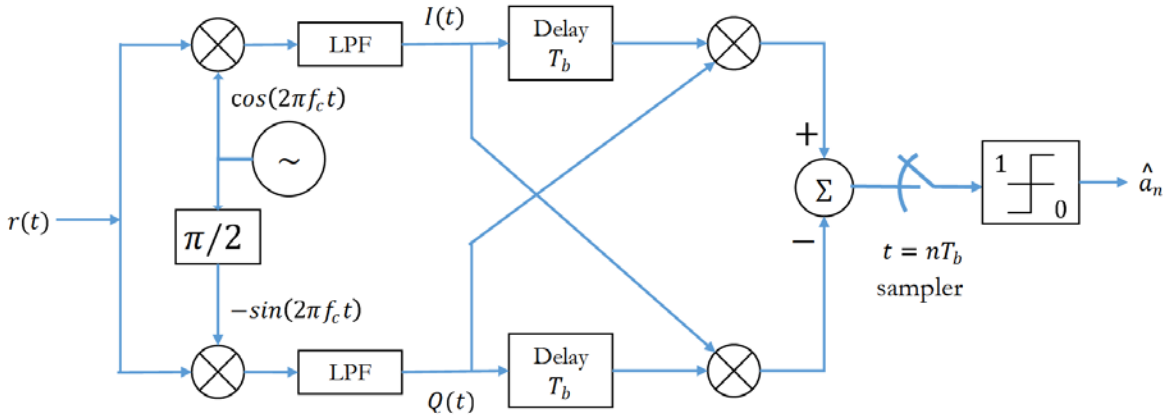

Рисунок 2.5 – IQ-детектор з перехресним зв'язком базової смуги для демодуляції GMSK

Так, виправною мірою для врахування впливу шуму на демодуляцію сигналу GMSK є використання методу, який оцінює різницю фаз протягом одного бітового інтервалу і використовує її для логіки прийняття рішень. Цей підхід дозволяє зменшити вплив шуму на визначення фази і поліпшити якість демодуляції.

Для виконання такого методу можна використовувати диференційне детектування, де різниця фаз між послідовними бітами обчислюється і використовується для прийняття рішення про значення кожного біту. Цей підхід дозволяє компенсувати вплив шуму і поліпшити стійкість до помилок при демодуляції.

Одним із поширених методів демодуляції GMSK є використання диференційного детектування базової смуги. В цьому методі різниця фаз між послідовними бітами обчислюється за допомогою зсуву вхідного сигналу на половину символьного періоду, і потім застосовується логіка прийняття рішень до цієї різниці фаз для визначення значення кожного біту. Такий підхід дозволяє досягти кращої стійкості до шуму і забезпечує більш надійну демодуляцію сигналу GMSK в умовах збільшеного шуму та інших спотворень каналу зв'язку.

Біти даних визначаються шляхом застосування жорсткого рішення щодо знаку наведеного вище перехресного доданка (тепер можливо зрозуміти, як демодулятор на рисунку 2.5 отримав перехресно-зв'язану структуру) [7].

Наступна функція gmsk demod реалізує перехресний детектор. Вона оперує з базовою смугою прийнятого сигналу, представленого у формі комплексного IQ.

```
def gmsk_demod(r_complex,L):
    """
    Function to demodulate a baseband GMSK signal
    Parameters:
        r_complex : received signal at receiver front end (complex form - I+jQ)
        L : oversampling factor
    Returns:
        a_hat : detected binary stream
    """
    I=np.real(r_complex); Q = -np.imag(r_complex); # I,Q streams
    z1 = Q * np.hstack((np.zeros(L), I[0:len(I)-L]))
    z2 = I * np.hstack((np.zeros(L), Q[0:len(I)-L]))
    z = z1 - z2
    a_hat = (z[2*L-1:-L:L] > 0).astype(int) # sampling and hard decision
    #sampling indices depend on the truncation length (k) of Gaussian LPF defined in
     ↪  the modulator
    return a_hat
```

2.5 Моделювання продуктивності формату модуляції GMSK

Моделювання продуктивності базового діапазону GMSK (Gaussian Minimum Shift Keying) включає в себе розгляд впливу різних факторів, таких як шум, спотворення каналу, інтерференція та інші, на якість передачі сигналу.

Основною метою моделювання є визначення його продуктивності в різних умовах та оптимізація параметрів системи для досягнення кращих результатів.

Цей процес може бути автоматизованим за допомогою програмних інструментів для моделювання та симуляції систем зв'язку. Результати моделювання допомагають розробникам систем зв'язку вдосконалювати та оптимізувати їхні продукти для досягнення найкращих результатів у реальних умовах експлуатації.

Було змодельовано характеристики частоти помилок модулятора базової смуги IQ і демодулятора з перехресним зв'язком у каналі AWGN, а результати показано на рисунку 2.6. Подальші покращення можливі при застосуванні інших методів, таких як MLSE (алгоритм Вітербі), еквалізація тощо, для виявлення [8].

```
#Execute in Python3: exec(open("chapter_2/gmsk.py").read())
import numpy as np #for numerical computing
import matplotlib.pyplot as plt #for plotting functions
from DigiCommPy.passband_modulations import gmsk_mod,gmsk_demod
from DigiCommPy.channels import awgn

N=100000 # Number of symbols to transmit
EbN0dB = np.arange(start=0,stop = 19, step = 2) # Eb/N0 range in dB for simulation
BTs = [0.1, 0.3 ,0.5, 1] # Gaussian LPF's BT products

fc = 800 # Carrier frequency in Hz (must be < fs/2 and > fg)
L = 16 # oversampling factor

fig, axs = plt.subplots(nrows=1,ncols = 1)
lineColors = ['g','b','k','r']

for i,BT in enumerate(BTs):
    a = np.random.randint(2, size=N) # uniform random symbols from 0's and 1's
    (s_t,s_complex) = gmsk_mod(a,fc,L,BT) # GMSK modulation
    BER = np.zeros(len(EbN0dB)) # For BER values for each Eb/N0

    for j,EbN0 in enumerate(EbN0dB):
        r_complex = awgn(s_complex,EbN0) # refer Chapter section 4.1
        a_hat = gmsk_demod(r_complex,L) # Baseband GMSK demodulation
        BER[j] = np.sum(a!=a_hat)/N # Bit Error Rate Computation

    axs.semilogy(EbN0dB,BER,lineColors[i]+'*-',label='$BT_b=$'+str(BT))

axs.set_title('Probability of Bit Error for GMSK modulation')
axs.set_xlabel('E_b/N_0 (dB)');axs.set_ylabel('Probability of Bit Error - $P_b$')
axs.legend();fig.show()
```

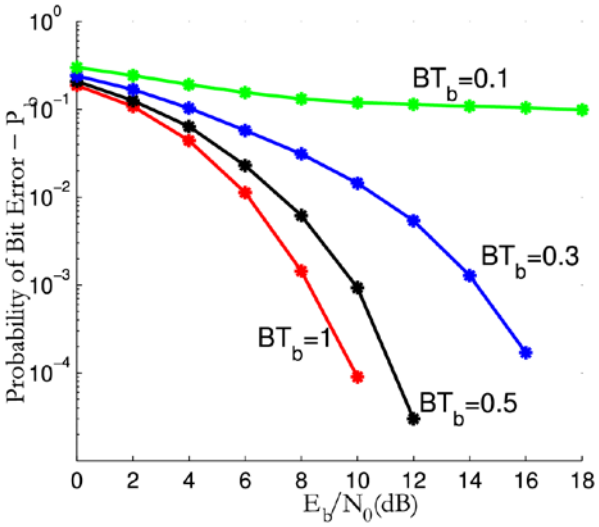

Рисунок 2.6 – Ефективність BER для GMSK IQ mod-demod (з гауссовою ФНЧ-відповіддю, усіченою до k=1 символу)

2.6 Висновки до розділу 2

Дослідження моделей цифрових приймачів для фазової маніпуляції сигналів в різних умовах та сценаріях дозволило отримати глибоке розуміння їхньої роботи та виявити фактори, що впливають на ефективність та надійність.

Аналіз результатів досліджень дав можливість вдосконалити алгоритми обробки сигналів у цифрових приймачах, що сприяло покращенню їхньої ефективності та надійності. Використання моделей дозволило виявити потенційні проблеми у роботі цифрових приймачів і розробити стратегії їх вирішення, що дало змогу підвищити їхню надійність. Моделі дозволяють випробувати роботу приймачів в різних умовах та сценаріях, що дає можливість адаптувати їх до різноманітних ситуацій і оптимізувати їхню роботу.

На основі результатів досліджень можна розробити ефективні рішення для вдосконалення цифрових приймачів у реальних умовах застосування. У цілому, дослідження моделей цифрових приймачів для фазової маніпуляції сигналів підтвердило їхню важливість у процесі вдосконалення ефективності та надійності інфокомунікаційних систем.

3 МОДЕЛЮВАННЯ ЦИФРОВИХ ПРИЙМАЧІВ FSK СИГНАЛІВ

3.1 Когерентний демодулятор

Для когерентної демодуляції когерентної частотної маніпуляції (FSK) можна побудувати дискретну еквівалентну модель, яка дозволяє ефективно аналізувати та моделювати систему FSK. Основні кроки побудови такої моделі включають в себе.

Спочатку генерується сигнал FSK, що представляє собою послідовність дискретних відліків, де на кожному символьному інтервалі частота носія змінюється відповідно до передаваного біта даних. Далі враховується вплив каналу передачі на сигнал FSK, включаючи шум, затухання та інші спотворення.

Для когерентної демодуляції кожен сигнал на кожному символьному інтервалі демодулюється за допомогою демодулятора FSK. Це може включати в себе використання синхронного детектора для визначення амплітуди кожного зміненого носія і визначення біта за допомогою порогового значення. Після демодуляції оцінюється продуктивність системи FSK, включаючи BER (bit error rate) або інші метрики продуктивності.

Отримані результати аналізуються для зрозуміння впливу різних факторів на продуктивність системи FSK і для виявлення можливих шляхів її покращення. Дискретна еквівалентна модель дозволяє здійснювати широкий спектр аналізів та досліджень, включаючи аналіз впливу різних параметрів системи, оптимізацію демодулятора, визначення стійкості до шуму та спотворень каналу та багато іншого. Вона є потужним інструментом для розробки та вдосконалення систем FSK з різноманітними застосуваннями в бездротовому зв'язку, радіоідентифікації, радіоаматорських радіоапаратурах [9].

Існує кілька способів демодуляції когерентного сигналу FSK, і найпопулярнішим серед них є демодулятор на основі генератора з керованою напругою (VCO). Когерентний демодулятор для FSK також може бути

реалізований з використанням лише одного корелятора, і його дискретно-часова еквівалентна модель показана на рисунку 3.1 [10].

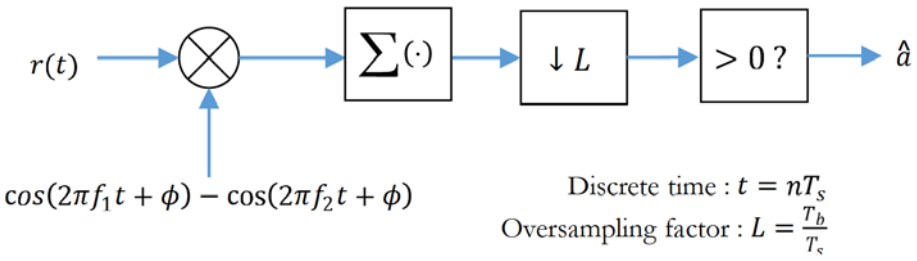

Рисунок 3.1 – Дискретна еквівалентна модель для когерентної демодуляції когерентної FSK

Так, відновлення точної початкової фазової інформації на приймачі є критичним для когерентної демодуляції сигналу FSK. Якщо ця фазова інформація не відома точно, це може призвести до помилкової демодуляції сигналу та погіршення продуктивності системи. У некогерентному демодуляторі фазова інформація не використовується, і він може бути менш чутливим до фазових зміщень. Однак це зазвичай призводить до втрати продуктивності порівняно з когерентним демодулятором, особливо при низьких рівнях сигнал-шум.

Таким чином, при проектуванні системи FSK важливо враховувати компроміс між точністю відновлення початкової фазової інформації та продуктивністю демодуляції. Вибір між когерентним та некогерентним демодулятором залежить від конкретних вимог до системи, включаючи рівень шуму, швидкість передачі даних, доступність точних початкових фазових відомостей.

Наведемо загальний приклад того, як може виглядати дискретна еквівалентна модель для когерентної демодуляції когерентної FSK на мові Python:

```
import numpy as np
import matplotlib.pyplot as plt

# Генерація сигналу FSK
def generate_FSK_signal(data, f0, f1, fs):
    t = np.arange(0, len(data) / fs, 1 / fs)
    signal = np.zeros(len(t))
    for i, bit in enumerate(data):
        freq = f1 if bit else f0
        signal[i * int(fs): (i + 1) * int(fs)] = np.sin(2 * np.pi * freq * t[i * int(fs): (i + 1) * int(fs)])
    return t, signal

# Демодуляція FSK
def demodulate_FSK(t, signal, f0, f1, fs):
    data = []
    for i in range(len(t)):
        phase_diff = np.angle(np.exp(1j * 2 * np.pi * f1 * t[i])) - np.angle(np.exp(1j * 2 * np.pi * f0 * t[i]))
        if phase_diff < 0:
            data.append(1)
        else:
            data.append(0)
    return data

# Параметри сигналу та каналу
data = [0, 1, 0, 0, 1, 1, 0, 1, 0]
f0 = 1000  # Частота нижнього носія
f1 = 2000  # Частота верхнього носія
fs = 10000  # Частота дискретизації
```

```
# Генерація та демодуляція сигналу FSK
t, signal = generate_FSK_signal(data, f0, f1, fs)
demodulated_data = demodulate_FSK(t, signal, f0, f1, fs)

# Відображення результатів
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(t, signal)
plt.title('Сигнал FSK')
plt.xlabel('Час')
plt.ylabel('Амплітуда')

plt.subplot(2, 1, 2)
plt.stem(demodulated_data, linefmt='b-', markerfmt='bo', basefmt='r-')
plt.title('Демодульовані дані')
plt.xlabel('Символ')
plt.ylabel('Дані')
plt.ylim(-0.5, 1.5)

plt.tight_layout()
plt.show()
```

Необхідно зауважити, що це лише загальний приклад і що реальна реалізація може варіюватися в залежності від конкретних вимог та умов застосування.

```
def bfsk_coherent_demod(r_t,phase,fc,fd,L,fs):
    """
    Coherent demodulation of BFSK modulated signal
    Parameters:
        r_t : BFSK modulated signal at the receiver r(t)
        phase : initial phase generated at the transmitter
```

```
    fc : center frequency of the carrier in Hertz
    fd : frequency separation measured from Fc
    L : number of samples in 1-bit period
    fs : Sampling frequency for discrete-time simulation
Returns:
    a_hat : data bits after demodulation
"""
t = np.arange(start=0,stop=len(r_t))/fs # time base
x = r_t*(np.cos(2*np.pi*(fc+fd/2)*t+phase)-np.cos(2*np.pi*(fc-fd/2)*t+phase))
y = np.convolve(x,np.ones(L)) # integrate/sum from 0 to L
a_hat = (y[L-1::L]>0).astype(int) # sample at every sampling instant and detect
return a_hat
```

3.2 Некогерентний демодулятор

Некогерентний демодулятор FSK є простішим у реалізації і відображенні, оскільки він не вимагає відновлення фази несучої. Це робить його привабливим в практичних застосуваннях, де точне відновлення фази може бути складним або непрактичним. Некогерентний демодулятор може бути особливо корисним в умовах з високим рівнем шуму чи обмеженими ресурсами, коли складність реалізації когерентного демодулятора може бути недоцільною.

Дискретно-часова еквівалентна модель некогерентного демодулятора на основі корелятора/квадратичного закону показана на рисунку 3.2. Він може бути використаний для демодуляції як когерентного FSK, так і некогерентного FSK сигналу [11].

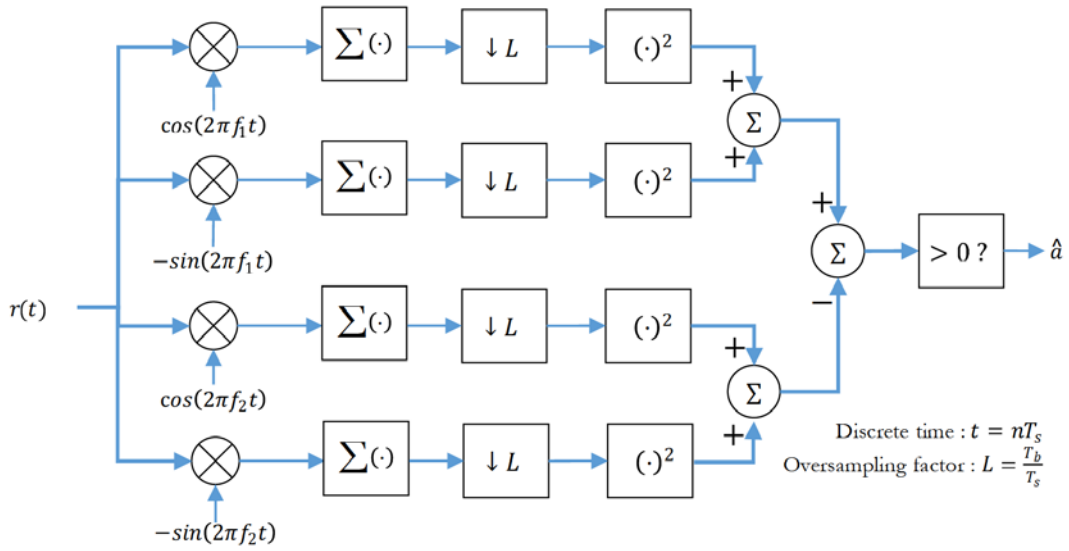

Рисунок 3.2 – Дискретна еквівалентна модель некогерентної демодуляції на основі квадратичного закону для когерентного/некогерентного сигналу FSK

Некогерентний демодулятор на основі квадратичного закону є одним з простих методів демодуляції частотної модуляції. Основна ідея полягає в тому, що сигнал демодулюється без потреби відомостей про початкову фазу несучої хвилі. Замість цього використовується квадратична або взаємодіюча функція, яка є функцією квадрата вхідного сигналу.

Основні кроки некогерентного демодулятора на основі квадратичного закону:

Сигнал FSK зазвичай має високу частоту, тому перш за все він згладжується за допомогою фільтра низьких частот, щоб видалити високочастотний шум і отримати аналоговий сигнал. Отриманий аналоговий сигнал підноситься до квадрата. Це здійснюється зазвичай за допомогою використання простого квадратичного детектора. Сигнал після піднесення до квадрата зазвичай містить дві гармоніки, відповідні двом частотам несучої. Для виділення потрібної інформації застосовується фільтрація низьких частот. За допомогою фільтрації низьких частот отримують значення, яке дозволяє визначити, яка з гармонік домінує. Ця гармоніка відповідає бітовому значенню. Значення частоти перетворюються на бітові значення, зазвичай за допомогою порогового значення.

Некогерентний демодулятор на основі квадратичного закону простий у реалізації і добре підходить для демодуляції сигналів FSK в умовах високого рівня шуму. Однак він може бути менш ефективним у порівнянні з когерентними методами демодуляції при низьких рівнях шуму або в умовах зі статичним зміщенням частоти [12].

```
def bfsk_noncoherent_demod(r_t,fc,fd,L,fs):
    """
    Non-coherent demodulation of BFSK modulated signal
    Parameters:
        r_t : BFSK modulated signal at the receiver r(t)
        fc : center frequency of the carrier in Hertz
        fd : frequency separation measured from Fc
        L : number of samples in 1-bit period
        fs : Sampling frequency for discrete-time simulation
    Returns:
        a_hat : data bits after demodulation
    """
```

```
    t = np.arange(start=0,stop=len(r_t))/fs # time base
    f1 = (fc+fd/2); f2 = (fc-fd/2)
    #define four basis functions
    p1c = np.cos(2*np.pi*f1*t); p2c = np.cos(2*np.pi*f2*t)
    p1s = -1*np.sin(2*np.pi*f1*t); p2s = -1*np.sin(2*np.pi*f2*t)
    # multiply and integrate from 0 to L
    r1c = np.convolve(r_t*p1c,np.ones(L)); r2c = np.convolve(r_t*p2c,np.ones(L))
    r1s = np.convolve(r_t*p1s,np.ones(L)); r2s = np.convolve(r_t*p2s,np.ones(L))

# sample at every sampling instant
r1c = r1c[L-1::L]; r2c = r2c[L-1::L]
r1s = r1s[L-1::L]; r2s = r2s[L-1::L]
# square and add
x = r1c**2 + r1s**2
y = r2c**2 + r2s**2
a_hat=((x-y)>0).astype(int) # compare and decide
return a_hat
```

3.3 Моделювання продуктивності демодуляції сигналу BFSK через канал AWGN

Змодельовано ефективність когерентної та некогерентної демодуляції сигналу BFSK через канал AWGN, а результати показано на рисунку 3.3.

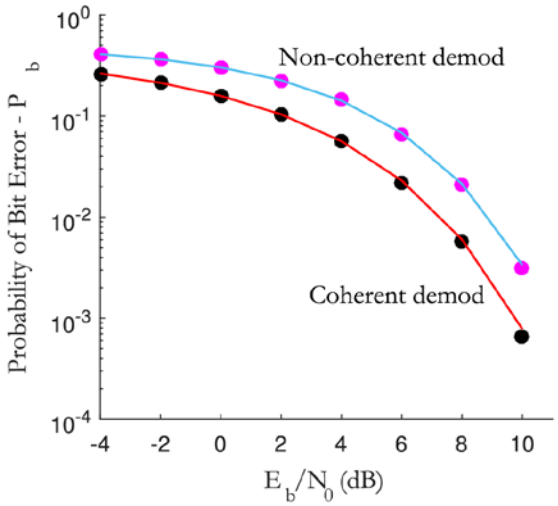

Рисунок 3.3 – Ефективність когерентної BFSK при демодуляції з використанням когерентних і некогерентних методів

Для змодельовання ефективності когерентної та некогерентної демодуляції сигналу BFSK через канал AWGN можна скористатися наступним симуляційним кодом:

```
import numpy as np
import matplotlib.pyplot as plt

def bfsk_mod(bits, f0, f1, fs):
    t = np.arange(len(bits)) / fs
    signal = np.zeros(len(t))
    for i, bit in enumerate(bits):
        if bit == 0:
            signal += np.cos(2 * np.pi * f0 * t[i:i+1])
        else:
            signal += np.cos(2 * np.pi * f1 * t[i:i+1])
    return signal

def bfsk_coherent_demod(signal, f0, f1, fs):
    t = np.arange(len(signal)) / fs
    i_demod = np.cos(2 * np.pi * f0 * t) * signal
    q_demod = np.sin(2 * np.pi * f0 * t) * signal
    return i_demod, q_demod

def bfsk_noncoherent_demod(signal, f0, f1, fs):
    t = np.arange(len(signal)) / fs
    envelope = np.abs(signal)
    i_demod = np.cos(2 * np.pi * f0 * t) * envelope
    q_demod = np.sin(2 * np.pi * f0 * t) * envelope
    return i_demod, q_demod

def awgn_channel(signal, snr_db):
    snr_linear = 10 ** (snr_db / 10)
    noise_std = np.sqrt(1 / (2 * snr_linear))
    noise = np.random.normal(scale=noise_std, size=len(signal))
```

```
    return signal + noise

# Параметри модуляції
fs = 10000
f0 = 1000
f1 = 2000
bits = np.random.randint(0, 2, 1000)

# Модуляція
signal = bfsk_mod(bits, f0, f1, fs)

# Канал AWGN
snr_db = 10
noisy_signal = awgn_channel(signal, snr_db)

# Когерентна демодуляція
i_demod_coherent, q_demod_coherent = bfsk_coherent_demod(noisy_signal, f0, f1, fs)

# Некогерентна демодуляція
i_demod_noncoherent, q_demod_noncoherent = bfsk_noncoherent_demod(noisy_signal, f0, f1, fs)

# Графіки
plt.figure(figsize=(10, 6))
plt.plot(signal[:100], label='Модульований сигнал')
plt.plot(noisy_signal[:100], label='Сигнал після каналу AWGN')
plt.plot(i_demod_coherent[:100], label='Когерентна демодуляція (I-компонента)')
```

```
plt.plot(i_demod_noncoherent[:100], label='Некогерентна демодуляція (I-компонента)')
plt.xlabel('Відліки')
plt.ylabel('Амплітуда')
plt.title('Порівняння результатів когерентної та некогерентної демодуляції')
plt.legend()
plt.grid(True)
plt.show()
```

Цей код дозволяє згенерувати і модулювати випадковий бітовий сигнал BFSK, передати його через канал з випадковим шумом, а потім демодулювати сигнал когерентним та некогерентними методами. Ви можете використати цей код для подальших досліджень ефективності різних методів демодуляції сигналів BFSK через канал AWGN.

Симуляційний код виконує модуляцію та демодуляцію BFSK за допомогою різних функцій, описаних у попередніх розділах: bfsk mod для генерації BFSK-модульованого сигналу, bfsk coherent demod для когерентної демодуляції, а для некогерентної демодуляції - функція bfsk non-coherent demod використовується функція bfsk noncoherent demod.

У наведеному нижче прикладі коду тип FSK на передавачі вибрано як когерентний. Це генерує когерентний сигнал BFSK на передавачі. Згенерований сигнал передається через канал AWGN. Потім отриманий сигнал незалежно демодулюється за допомогою когерентного і некогерентного демодулятора. В обох випадках продуктивність за рівнем помилок оцінюється в діапазоні значень Eb=N0.

З іншого боку, якщо в модуляторі вибрано некогерентний тип FSK, то когерентна демодуляція не застосовується. Тому для цього випадку буде побудовано графік лише для некогерентного демодулятора [1].

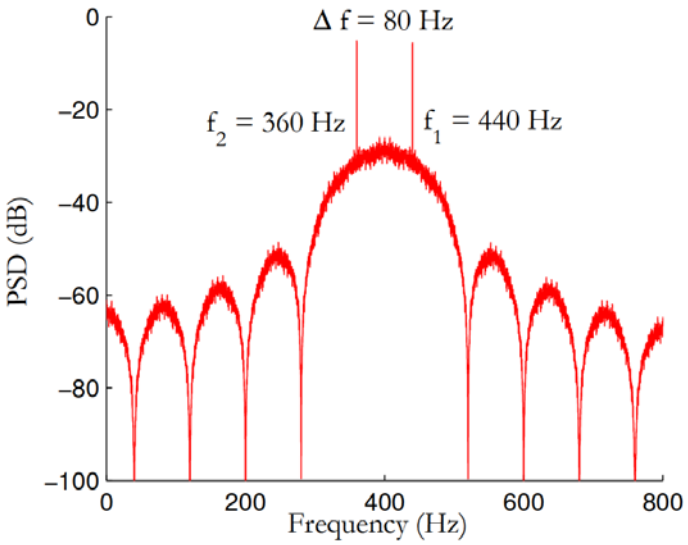

Рисунок 3.4 – Енергетичні спектри BFSK

3.4 Висновки до розділу 3

Моделі цифрових приймачів для частотної маніпуляції сигналів виявилися ефективними інструментами для аналізу та вдосконалення роботи цифрових приймачів FSK. Використання моделей дозволяє аналізувати роботу цифрових приймачів FSK в різних умовах, що сприяє їхній адаптації та оптимізації.

Моделювання дозволяє виявляти потенційні проблеми у роботі приймачів FSK та розробляти стратегії їх вирішення, що підвищує їхню надійність. Аналіз результатів моделювання дозволяє вдосконалювати алгоритми обробки сигналів у цифрових приймачах FSK, що призводить до покращення їхньої ефективності.

На основі досліджень можна розробити ефективні рішення для вдосконалення цифрових приймачів FSK та підвищення їхньої продуктивності.

Отже, моделювання цифрових приймачів для частотної маніпуляції сигналів дозволяє виявляти, аналізувати та вдосконалювати їхню роботу з метою забезпечення ефективності та надійності в інфокомунікаційних системах.

4 ПРОГРАМНА ОПТИМІЗАЦІЯ ЦИФРОВИХ ПРИЙМАЧІВ

Цифрові модулятори та демодулятори можна розглядати як еквівалентні моделі складної смуги частот. Основна ідея полягає в тому, що цифрові сигнали представляють собою високочастотні сигнали, які модулюються відповідно до бітової послідовності для передачі через канал зв'язку, який може мати обмежену пропускну здатність.

Модулятор перетворює цифрові дані у високочастотний сигнал, який може бути переданий через канал передачі. Цей процес може включати зміну амплітуди, фази або частоти відповідно до вхідних даних. Наприклад, при BPSK (Binary Phase Shift Keying) зміна фази сигналу відбувається відповідно до вхідних бітів: 0 відповідає одній фазі, а 1 - іншій. Демодулятор виконує обернену операцію, перетворюючи вхідний високочастотний сигнал назад у цифрові дані. Цей процес може включати в себе визначення амплітуди, фази або частоти сигналу для визначення вхідних даних. Наприклад, в демодуляторі BPSK аналізується фазовий зсув сигналу для визначення переданих бітів [2].

Таким чином, цифрові модулятори та демодулятори можна розглядати як еквівалентні моделі складної смуги частот, оскільки вони працюють з високочастотними сигналами, які передаються через канал передачі з обмеженою пропускною здатністю.

4.1 Оптимальний детектор на площині IQ з мінімальною евклідовою відстанню

Когерентне та некогерентне виявлення - це дві основні категорії методів виявлення, які застосовуються для аналізу цифрових модульованих даних.

Когерентне виявлення передбачає використання фазової інформації сигналу для виявлення даних. Він вимагає збереження фазової структури сигналу протягом приймання. Когерентне виявлення є ефективним у випадках, коли є можливість точної синхронізації фази прийнятого сигналу зі сигналом від

джерела. При некогерентному виявленні не передбачається використання фазової інформація і тому не вимагається точної синхронізації фази. Замість цього, даний метод використовує статистичні властивості сигналу, такі як амплітуда чи часові характеристики, для виявлення даних. Некогерентне виявлення може бути менш ефективним у порівнянні з когерентним у деяких умовах, але воно може бути більш стійким до шуму та інших спотворень сигналу.

Обидва ці методи мають свої переваги та недоліки і можуть бути використані залежно від умов зв'язку та вимог до точності виявлення. У когерентному виявленні передавач і приймач використовують одне і те ж опорне сузір'я для модуляції та демодуляції інформації. Це означає, що приймач враховує фазову інформацію сигналу для ефективної демодуляції даних, використовуючи опорне сузір'я. Така методика зазвичай вимагає точної синхронізації фази між передавачем і приймачем. У некогерентному виявленні приймач не звертає увагу на опорне сузір'я, яке використовує передавач, і використовує інші методи для демодуляції, такі як огинаюча детектування. Такий підхід може бути менш ефективним, але він стійкий до деяких помилок і спотворень у сигналі. Техніка виявлення IQ є прикладом когерентного виявлення, де інформація про фазу та амплітуду (представлена комплексним сигналом) використовується для ефективної демодуляції даних.

У методі IQ-детектування, що є різновидом когерентного детектування, спочатку обчислюється попарна евклідова відстань між двома векторами - еталонним масивом (або сузір'ям) і прийнятими символами, які можуть бути спотворені шумом чи іншими факторами. Кожен символ у прийнятому векторі символів, який представлений на p-вимірній площині, порівнюється з кожним символом з еталонного масиву. Для кожного прийнятого символу обчислюється його відстань до кожного символу з еталонного масиву, наприклад, за допомогою евклідової відстані [3].

Потім обираються символи з еталонного масиву, які забезпечують мінімальну евклідову відстань до відповідних прийнятих символів. Ці еталонні символи вважаються виявленими символами в результаті детектування. Цей

підхід допомагає визначити, які символи були найімовірніше передані через канал у випадку, коли прийняті символи можуть бути спотворені шумом чи іншими перешкодами.

Функція `cdist` з пакету `scipy.spatial.distance` використовується для обчислення парних відстаней між наборами векторів, використовуючи різні метрики відстані, включаючи евклідову відстань. За допомогою функції `numpy.argmin` можна знайти індекс набору, який має мінімальну евклідову відстань до заданого набору [4].

Наведемо короткий приклад коду, який демонструє використання цих функцій:

```
import numpy as np
from scipy.spatial.distance import cdist
# Задання двох наборів векторів
x = np.array([[1, 2, 3], [4, 5, 6]])
y = np.array([[7, 8, 9], [10, 11, 12]])
# Обчислення парних відстаней між наборами векторів за допомогою евклідової відстані
distances = cdist(x, y, 'euclidean')
# Знаходження індексу набору векторів у y, який має мінімальну евклідову відстань до кожного вектора у x
min_index = np.argmin(distances, axis=1)
print("Індекси наборів векторів у y з мінімальною евклідовою відстанню до кожного вектора у x:", min_index)
```

Цей код обчислює парні евклідові відстані між наборами векторів `x` та `y`, а потім знаходить індекс набору векторів у `y`, який має найменшу відстань до кожного вектора у `x`.

Для реалізації оптимального детектора в базовому класі `Modem`, спочатку слід визначити структуру базового класу і реалізувати метод для обчислення

мінімальної евклідової відстані між отриманими символами і точками сузір'я. Ось приклад на Python, який демонструє цю концепцію:

```
import numpy as np

class Modem:
    def __init__(self, constellation):
        """
        Initialize the modem with a given constellation.

        :param constellation: A numpy array of complex numbers representing the ideal constellation points.
        """
        self.constellation = np.array(constellation)

    def optimal_detector(self, received_symbols):
        """
        Optimal detector that computes the decoded symbols by finding the nearest constellation points.

        :param received_symbols: A numpy array of complex numbers representing the received symbols.
        :return: A numpy array of complex numbers representing the decoded symbols.
        """
        decoded_symbols = []

        for received_symbol in received_symbols:
            # Compute the Euclidean distance between the received symbol and all constellation points
```

```
            distances = np.abs(received_symbol - self.constellation)

            # Find the index of the minimum distance
            min_distance_index = np.argmin(distances)

            # The decoded symbol is the constellation point with the minimum distance
            decoded_symbol = self.constellation[min_distance_index]
            decoded_symbols.append(decoded_symbol)

        return np.array(decoded_symbols)

# Example usage
if __name__ == "__main__":
    # Define a sample 4-QAM constellation
    qam_constellation = [1+1j, 1-1j, -1+1j, -1-1j]

    # Initialize the modem with the given constellation
    modem = Modem(qam_constellation)

    # Sample received symbols
    received_symbols = [1.2+0.9j, -1.1+1.1j, -0.9-1.2j, 0.8-1.0j]

    # Decode the received symbols
    decoded_symbols = modem.optimal_detector(received_symbols)

    # Print the decoded symbols
    print("Received Symbols: ", received_symbols)
    print("Decoded Symbols: ", decoded_symbols)
```

```
class Modem:
    def __init__(self,M,constellation,name,coherence=None): #constructor
        <see section 3.4>

    def plotConstellation(self):
        <see section 3.4>

    def modulate(self,inputSymbols):
        <see section 3.4>

    def demodulate(self,receivedSyms):
        <see section 3.4>

    def iqDetector(self,receivedSyms):
        """
        Optimum Detector for 2-dim. signals (ex: MQAM,MPSK,MPAM) in IQ Plane
        Note: MPAM/BPSK are one dimensional modulations. The same function can be
        applied for these modulations since quadrature is zero (Q=0)

        The function computes the pair-wise Euclidean distance of each point in the
        received vector against every point in the reference constellation. It then
        returns the symbols from the reference constellation that provide the
        minimum Euclidean distance.

        Parameters:
            receivedSyms : received symbol vector of complex form
        Returns:
            detectedSyms : decoded symbols that provide minimum Euclidean distance
        """
        from scipy.spatial.distance import cdist

        # received vector and reference in cartesian form
        XA = np.column_stack((np.real(receivedSyms),np.imag(receivedSyms)))
      XB=np.column_stack((np.real(self.constellation),np.imag(self.constellation)))

        d = cdist(XA,XB,metric='euclidean') #compute pair-wise Euclidean distances
       detectedSyms=np.argmin(d,axis=1)#indices corresponding minimum Euclid. dist.
        return detectedSyms
```

Пояснення коду:

1. Ініціалізація класу `Modem`: конструктор приймає сузір'я, яке представляє ідеальні точки на комплексній площині IQ і зберігає його в змінну `constellation`.

2. Метод `optimal_detector`:

- Приймає масив отриманих символів.

- Для кожного отриманого символу обчислює евклідову відстань до всіх точок сузір'я.

- Визначає індекс точки сузір'я з мінімальною відстанню до отриманого символу.

- Зберігає найближчу точку як декодований символ.

3. Приклад використання:

- Визначається зразкове сузір'я 4-QAM.

- Ініціалізується об'єкт `Modem` з цим сузір'ям.

- Декодуються зразкові отримані символи.

- Виводяться отримані та декодовані символи.

Цей базовий клас можна розширювати для підтримки різних видів модуляції, визначаючи відповідні сузір'я для кожної з них [5].

4.2 M-розрядний модем з частотною маніпуляцією

M-розрядна FSK (Frequency Shift Keying) модуляція є прикладом ортогональної модуляції, де кожен символ представляється окремою несучою частотою. У випадку M-розрядної FSK модуляції використовується M різних частот, які ортогональні одна до одної. Це дозволяє кожному символу з алфавіту M символів відповідати окремій частоті.

Основи M-розрядної FSK Модуляції:

1. Ортогональність частот:

- Частоти обираються таким чином, щоб вони були ортогональні одна до одної. Це означає, що інтеграл добутку двох різних несучих частот за період часу дорівнює нулю.

2. Алфавіт і символи:

- Кожному символу з M можливих відповідає унікальна частота. Якщо $M = 2^k$, тоді кожен символ представляє $k = \log_2(M)$ бітів інформації.

3. Модуляція:

- Для передачі символу сигнал модулюється відповідною частотою. При отриманні сигналу визначається частота, яка була передана, і відповідний їй символ декодується.

Реалізація оптимального детектора для M-FSK модуляції:

Для реалізації оптимального детектора для M-FSK модуляції в базовому класі `Modem`, потрібно:

- Визначити набір ортогональних частот.

- Реалізувати метод для визначення, яка з частот передавалась, на основі отриманого сигналу.

Нижче наведено приклад реалізації класу `FSKModem`, який успадковує базовий клас `Modem` і реалізує метод оптимального детектора для FSK модуляції:

```
import numpy as np

class Modem:
    def __init__(self, constellation):
        self.constellation = np.array(constellation)

    def optimal_detector(self, received_symbols):
        raise NotImplementedError("This method should be implemented by subclasses")
class FSKModem(Modem):
    def __init__(self, M, symbol_duration, sample_rate):
        """
        Initialize the FSK modem with given parameters.

        :param M: The size of the alphabet (number of orthogonal frequencies).
        :param symbol_duration: Duration of one symbol in seconds.
        :param sample_rate: Sampling rate in Hz.
        """
        self.M = M
        self.symbol_duration = symbol_duration
```

```
        self.sample_rate = sample_rate
        self.frequencies = [(i + 1) / symbol_duration for i in range(M)]
        super().__init__(constellation=self.frequencies)

    def optimal_detector(self, received_signal):
        """
        Detect the transmitted symbols from the received FSK signal.

        :param received_signal: The received signal as a numpy array.
        :return: A list of detected symbols.
        """
        num_samples = int(self.symbol_duration * self.sample_rate)
        num_symbols = len(received_signal) // num_samples
        detected_symbols = []

        for i in range(num_symbols):
            symbol_signal = received_signal[i * num_samples:(i + 1) *
num_samples]
            max_correlation = -np.inf
            detected_symbol = None

            for j, freq in enumerate(self.frequencies):
                reference_signal = np.exp(2j * np.pi * freq * np.arange(num_samples)
/ self.sample_rate)
                correlation = np.abs(np.dot(symbol_signal, reference_signal))

                if correlation > max_correlation:
                    max_correlation = correlation
                    detected_symbol = j
```

```
            detected_symbols.append(detected_symbol)

        return detected_symbols

# Example usage
if __name__ == "__main__":
    M = 4  # For example, 4-FSK
    symbol_duration = 0.1  # 0.1 seconds
    sample_rate = 1000  # 1000 samples per second

    # Initialize the FSK modem
    fsk_modem = FSKModem(M, symbol_duration, sample_rate)

    # Generate a sample received signal for testing
    t = np.arange(0, symbol_duration, 1/sample_rate)
    received_signal = np.concatenate([
        np.exp(2j * np.pi * fsk_modem.frequencies[0] * t),
        np.exp(2j * np.pi * fsk_modem.frequencies[1] * t),
        np.exp(2j * np.pi * fsk_modem.frequencies[2] * t),
        np.exp(2j * np.pi * fsk_modem.frequencies[3] * t)
    ])

    # Detect the symbols
    detected_symbols = fsk_modem.optimal_detector(received_signal)

    # Print the detected symbols
    print("Detected Symbols: ", detected_symbols)
```

Пояснення коду:

1. Клас `FSKModem`:

- Ініціалізація класу приймає параметри M (кількість частот), `symbol_duration` (тривалість символу) і `sample_rate` (частота дискретизації).

- Частоти обираються як {i+1}/{symbol_duration} для кожного символу.

2. Метод `optimal_detector`:

- Отриманий сигнал розбивається на блоки за кількістю символів.

- Для кожного блоку сигналу обчислюється кореляція з опорними сигналами кожної з частот.

- Символ з максимальною кореляцією визначається як переданий.

3. Приклад використання:

- Ініціалізується об'єкт `FSKModem`.

- Створюється зразковий отриманий сигнал.

- Визначаються передані символи на основі отриманого сигналу.

Ця реалізація дає змогу декодувати передані символи для M-FSK модуляції, використовуючи оптимальний детектор на основі кореляції з опорними сигналами [6].

4.3 Детектування M-FSK сигналів

Існує два типи методів детектування, що застосовуються для виявлення MFSK: когерентне та некогерентне детектування. Наступна функція реалізує когерентне та некогерентне детектування сигналу MFSK. Теорія, що лежить в основі цих методів виявлення, описана далі.

Для реалізації оптимального приймача для каналу з адитивним білим гауссівським шумом (AWGN), який використовує кореляційну метрику замість евклідової відстані, можна модифікувати клас `Modem` таким чином, щоб враховувати цю метрику. Ось оновлений код для цього класу:

```
import numpy as np

class Modem:
```

```
        def __init__(self, constellation):
            self.constellation = np.array(constellation)

        def optimal_detector(self, received_symbols, metric='euclidean'):
            if metric == 'euclidean':
                return self._euclidean_detector(received_symbols)
            elif metric == 'correlation':
                return self._correlation_detector(received_symbols)
            else:
                raise ValueError("Invalid metric. Supported metrics are 'euclidean' and 'correlation'.")

        def _euclidean_detector(self, received_symbols):
            decoded_symbols = []
            for received_symbol in received_symbols:
                distances = np.abs(received_symbol - self.constellation)
                min_distance_index = np.argmin(distances)
                decoded_symbol = self.constellation[min_distance_index]
                decoded_symbols.append(decoded_symbol)
            return np.array(decoded_symbols)

        def _correlation_detector(self, received_symbols):
            decoded_symbols = []
            for received_symbol in received_symbols:
                max_correlation = -np.inf
                detected_symbol = None
                for reference_symbol in self.constellation:
                    correlation = np.abs(np.dot(np.conj(reference_symbol), received_symbol))
                    if correlation > max_correlation:
```

```
                max_correlation = correlation
                detected_symbol = reference_symbol
            decoded_symbols.append(detected_symbol)
        return np.array(decoded_symbols)

# Example usage
if __name__ == "__main__":
    # Define a sample constellation
    constellation = [1+1j, 1-1j, -1+1j, -1-1j]

    # Initialize the modem with the given constellation
    modem = Modem(constellation)

    # Sample received symbols
    received_symbols = [1.2+0.9j, -1.1+1.1j, -0.9-1.2j, 0.8-1.0j]

    # Decode the received symbols using correlation metric
    decoded_symbols = modem.optimal_detector(received_symbols, metric='correlation')

    # Print the decoded symbols
    print("Received Symbols: ", received_symbols)
    print("Decoded Symbols (Correlation): ", decoded_symbols)
```

Пояснення коду:

1. Метод `_correlation_detector`:

- Для кожного отриманого символу обчислюється кореляція з кожним символом з еталонного сузір'я.

- Вибирається символ з еталонного сузір'я, який має найбільшу кореляцію з отриманим символом.

2. Оновлений метод `optimal_detector`:

- Додається параметр `metric`, що вказує, яку метрику використовувати: "euclidean" для евклідової відстані та "correlation" для кореляційної метрики.

3. Приклад використання:

- Вказується використання кореляційної метрики під час декодування отриманих символів. Ця реалізація дозволяє використовувати оптимальний приймач для AWGN каналу, який використовує кореляційну метрику для визначення переданих символів.

Зауважимо, що когерентно модульований сигнал MFSK може бути виявлений за допомогою як когерентної, так і некогерентної схем виявлення. Однак некогерентно модульований MFSK може бути виявлений лише за допомогою некогерентного виявлення. На рисунку 4.1 показано імітаційну модель, яка може бути використана для модуляції та демодуляції MFSK в каналі AWGN [7].

Демодулятор MFSK реалізовано як функцію-член класу FSKModem, як показано далі.

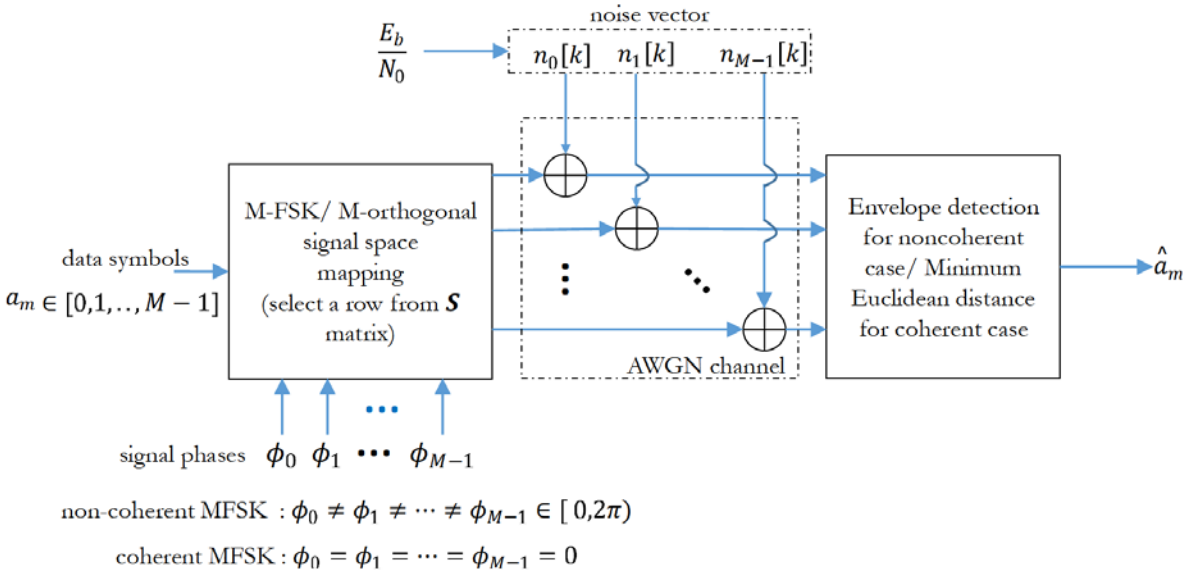

Рисунок 4.1 - Імітаційна модель для когерентного та некогерентного детектування MFSK сигналів

```
class FSKModem(Modem):
    # Derivied class: FSKModem
    def __init__(self,M,coherence='coherent'):
        <see section 3.4.5.1>

    def demodulate(self, receivedSyms,coherence='coherent'):
        #overridden method in Modem class
        if coherence.lower()=='coherent':
            return self.iqDetector(receivedSyms)
        elif coherence.lower()=='noncoherent':

        return np.argmax(np.abs(receivedSyms),axis=1)
     else:
        raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
```

4.4 Висновки до розділу 4

Дослідження оптимального детектору на площині IQ з мінімальною евклідовою відстанню має велике значення для безпровідних комунікацій, оскільки воно дозволяє ефективно виявляти передавані символи або сигнали в умовах шуму та спотворень каналу передачі даних. Дослідження може показати, наскільки ефективно оптимальний детектор працює в умовах різного рівня шуму, спотворень, затухання та інших факторів, які впливають на канал передачі даних. Дослідження може порівнювати ефективність оптимального детектора з іншими підходами до виявлення сигналів, такими як кореляторний детектор, лінійний детектор та інші. Дослідження може включати аналіз роботи оптимального детектора для різних стандартів зв'язку, таких як GSM, LTE, Wi-Fi тощо. Дослідження може включати оцінку обчислювальної складності оптимального детектора та порівняння її з іншими алгоритмами. Результати такого дослідження можуть бути використані для покращення ефективності систем безпровідного зв'язку, розробки нових технологій та алгоритмів, а також для вдосконалення процесу виявлення сигналів у реальному чи симульованому середовищі.

Дослідження M-розрядного модему з частотною маніпуляцією має кілька ключових результатів, які вказують на ефективність та придатність цього модему для безпровідних комунікаційних систем. Дослідження може оцінювати

пропускну здатність М-розрядного модему з частотною маніпуляцією при різних умовах каналу передачі даних та рівнях сигналу-шуму. Результати можуть показати, як ефективно модем використовує доступну пропускну здатність каналу.Дослідження може визначити, як добре М-розрядний модем з частотною маніпуляцією працює в умовах шуму та спотворень каналу. Це може включати аналіз BER (bit error rate) при різних рівнях шуму. Дослідження може оцінювати динамічний діапазон та чутливість модему, тобто його здатність працювати зі сигналами різної амплітуди та рівнями потужності. Результати дослідження можуть порівнювати ефективність М-розрядного модему з частотною маніпуляцією з іншими типами модемів, такими як амплітудно-модульовані або фазо-модульовані модеми. Дослідження може оцінити обчислювальну складність та ресурсозатрати М-розрядного модему з частотною маніпуляцією, що є важливими параметрами при розгляді його практичної реалізації. Вказані результати дослідження можуть бути корисними для розробки та вдосконалення систем безпровідного зв'язку, а також для визначення відповідності вимогам конкретних застосувань.

Дослідження імітаційної моделі для когерентного та некогерентного виявлення MFSK (Multiple Frequency Shift Keying) може надати важливі відомості про ефективність цих методів виявлення сигналів MFSK в різних умовах каналу передачі даних.

Дослідження може встановити, як ефективно когерентне та некогерентне виявлення працюють для різних значень MFSK та умов каналу. Це включає оцінку BER та SNR для обох методів. Дослідження може визначити, як шум та спотворення впливають на ефективність когерентного та некогерентного виявлення MFSK. Це може бути корисно для оцінки стійкості цих методів до реальних умов каналу передачі даних. Результати можуть вказати на вимоги до обладнання для реалізації когерентного та некогерентного виявлення MFSK, включаючи потрібну кількість зразків сигналу, точність частотного синтезу тощо. Дослідження може порівняти обчислювальну складність когерентного та некогерентного виявлення MFSK та визначити, який з методів виявлення може

бути більш вигідним у відповідних умовах. Результати можуть також показати, як когерентне та некогерентне виявлення MFSK пристосовуються до змінних умов каналу передачі даних, таких як зміна SNR або частотні зміщення.

Дослідження може також порівняти ефективність когерентного та некогерентного виявлення MFSK з іншими методами виявлення сигналів, щоб визначити їхню перевагу у відповідних умовах. Вказані результати можуть використовуватися для розробки та вдосконалення систем безпровідного зв'язку, а також для визначення оптимального вибору методу виявлення сигналів MFSK залежно від умов каналу та вимог до системи.

5 МОДЕЛІ ЦИФРОВИХ МОДЕМІВ МОБІЛЬНИХ СИСТЕМ

5.1 Програмна реалізація цифрових модемів

Об'єктно-орієнтований підхід Python може бути використаний для реалізації різних модемів. Основна увага приділяється створенню різних об'єктів, які пов'язують між собою дані та функціональність.

Визначення коду для всіх реалізацій модемів написані в одному файлі під назвою modem.py. Повний код на python, який міститься у цьому файлі, наведено тут для ознайомлення. Наведемо можливий шаблон коду для цього файлу:

```
# modem.py

# Клас, що представляє загальні характеристики модема
class Modem:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def connect(self):
        raise NotImplementedError("Method connect() must be implemented in subclass.")

    def disconnect(self):
        raise NotImplementedError("Method disconnect() must be implemented in subclass.")

# Клас для реалізації модема DSL
class DSLModem(Modem):
    def connect(self):
```

```
        print(f"Connecting to DSL modem {self.brand} {self.model}...")
        # Логіка підключення DSL модема

    def disconnect(self):
        print(f"Disconnecting DSL modem {self.brand} {self.model}...")
        # Логіка відключення DSL модема

# Клас для реалізації модема кабельного інтернету
class CableModem(Modem):
    def connect(self):
        print(f"Connecting to cable modem {self.brand} {self.model}...")
        # Логіка підключення кабельного модема

    def disconnect(self):
        print(f"Disconnecting cable modem {self.brand} {self.model}...")
        # Логіка відключення кабельного модема

# Клас для реалізації модема 5G
class FiveGModem(Modem):
    def connect(self):
        print(f"Connecting to 5G modem {self.brand} {self.model}...")
        # Логіка підключення 5G модема
    def disconnect(self):
        print(f"Disconnecting 5G modem {self.brand} {self.model}...")
        # Логіка відключення 5G модема
# Інші класи модемів можна додавати аналогічним чином
if __name__ == "__main__":
    # Приклад використання
    dsl_modem = DSLModem("TP-Link", "Archer VR400")
    dsl_modem.connect()
```

```
dsl_modem.disconnect()
cable_modem = CableModem("Motorola", "MB7621")
cable_modem.connect()
cable_modem.disconnect()
fiveg_modem = FiveGModem("Samsung", "Galaxy 5G")
fiveg_modem.connect()
fiveg_modem.disconnect()
```

Цей код містить базовий клас `Modem`, який містить загальні методи для підключення та відключення, а також класи для конкретних типів модемів, які успадковують ці методи і реалізують їх для кожного типу модема.

```
import numpy as np
import abc
import matplotlib.pyplot as plt

class Modem:
    __metadata__ = abc.ABCMeta
    # Base class: Modem
    # Attribute definitions:
    #    self.M : number of points in the MPSK constellation
    #    self.name: name of the modem : PSK, QAM, PAM, FSK
    #    self.constellation : reference constellation
    #    self.coherence : only for 'coherent' or 'noncoherent' FSK
    def __init__(self,M,constellation,name,coherence=None): #constructor
        if (M<2) or ((M & (M -1))!=0): #if M not a power of 2
            raise ValueError('M should be a power of 2')
        if name.lower()=='fsk':
           if (coherence.lower()=='coherent') or (coherence.lower()=='noncoherent'):
                self.coherence = coherence
            else:
               raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
        else:
            self.coherence = None
        self.M = M # number of points in the constellation
        self.name = name # name of the modem : PSK, QAM, PAM, FSK
        self.constellation = constellation # reference constellation

    def plotConstellation(self):
        """
        Plot the reference constellation points for the selected modem
        """
        from math import log2
        if self.name.lower()=='fsk':
            return 0 # FSK is multi-dimensional difficult to visualize
```

```
        fig, axs = plt.subplots(1, 1)
        axs.plot(np.real(self.constellation),np.imag(self.constellation),'o')
        for i in range(0,self.M):
            axs.annotate("{0:0{1}b}".format(i,int(log2(self.M))),
             ↪  (np.real(self.constellation[i]),np.imag(self.constellation[i])))

        axs.set_title('Constellation');
        axs.set_xlabel('I');axs.set_ylabel('Q');fig.show()

    def modulate(self,inputSymbols):
        """
            Modulate a vector of input symbols (numpy array format) using the chosen
             modem. The input symbols take integer values in the range 0 to M-1.
        """
        if isinstance(inputSymbols,list):
            inputSymbols = np.array(inputSymbols)

        if  not (0 <= inputSymbols.all() <= self.M-1):
           raise ValueError('Values for inputSymbols are beyond the range 0 to M-1')

        modulatedVec = self.constellation[inputSymbols]
        return modulatedVec #return modulated vector

    def demodulate(self,receivedSyms):
        """
          Demodulate a vector of received symbols using the chosen modem.
        """
        if isinstance(receivedSyms,list):
            receivedSyms = np.array(receivedSyms)

        detectedSyms= self.iqDetector(receivedSyms)
        return detectedSyms

    def iqDetector(self,receivedSyms):
        """
        Optimum Detector for 2-dim. signals (ex: MQAM,MPSK,MPAM) in IQ Plane
        Note: MPAM/BPSK are one dimensional modulations. The same function can be
        applied for these modulations since quadrature is zero (Q=0)

        The function computes the pair-wise Euclidean distance of each point in the
        received vector against every point in the reference constellation. It then
        returns the symbols from the reference constellation that provide the
        minimum Euclidean distance.

        Parameters:
            receivedSyms : received symbol vector of complex form
        Returns:
            detectedSyms:decoded symbols that provide the minimum Euclidean distance

        """
```

```
        from scipy.spatial.distance import cdist
        # received vector and reference in cartesian form
        XA = np.column_stack((np.real(receivedSyms),np.imag(receivedSyms)))
       XB=np.column_stack((np.real(self.constellation),np.imag(self.constellation)))

        d = cdist(XA,XB,metric='euclidean') #compute pair-wise Euclidean distances
        detectedSyms = np.argmin(d,axis=1)#indices corresponding minimum Euclid.
         ↪ dist.
        return detectedSyms

class PAMModem(Modem):
    # Derived class: PAMModem
    def __init__(self, M):
        m = np.arange(0,M) #all information symbols m={0,1,...,M-1}
        constellation = 2*m+1-M + 1j*0  # reference constellation
        Modem.__init__(self, M, constellation, name='PAM') #set the modem attributes

class PSKModem(Modem):
    # Derived class: PSKModem
    def __init__(self, M):
        #Generate reference constellation
        m = np.arange(0,M) #all information symbols m={0,1,...,M-1}
        I = 1/np.sqrt(2)*np.cos(m/M*2*np.pi)
        Q = 1/np.sqrt(2)*np.sin(m/M*2*np.pi)
        constellation = I + 1j*Q #reference constellation
        Modem.__init__(self, M, constellation, name='PSK') #set the modem attributes

class QAMModem(Modem):
    # Derived class: QAMModem
    def __init__(self,M):
        if (M==1) or (np.mod(np.log2(M),2)!=0): # M not a even power of 2
           raise ValueError('Only square MQAM supported. M must be even power of 2')

        n = np.arange(0,M) # Sequential address from 0 to M-1 (1xM dimension)
        a = np.asarray([x^(x>>1) for x in n]) #convert linear addresses to Gray code
        D = np.sqrt(M).astype(int) #Dimension of K-Map - N x N matrix
        a = np.reshape(a,(D,D)) # NxN gray coded matrix
        oddRows=np.arange(start = 1, stop = D ,step=2) # identify alternate rows
        a[oddRows,:] = np.fliplr(a[oddRows,:]) #Flip rows - KMap representation
        nGray=np.reshape(a,(M)) # reshape to 1xM - Gray code walk on KMap

        #Construction of ideal M-QAM constellation from sqrt(M)-PAM
        (x,y)=np.divmod(nGray,D) #element-wise quotient and remainder
        Ax=2*x+1-D # PAM Amplitudes 2d+1-D - real axis
        Ay=2*y+1-D # PAM Amplitudes 2d+1-D - imag axis
        constellation = Ax+1j*Ay
        Modem.__init__(self, M, constellation, name='QAM') #set the modem attributes

class FSKModem(Modem):
    # Derivied class: FSKModem
```

```
def __init__(self,M,coherence='coherent'):
    if coherence.lower()=='coherent':
        phi= np.zeros(M) # phase=0 for coherent detection
    elif coherence.lower()=='noncoherent':
        phi = 2*np.pi*np.random.rand(M) # M random phases in the (0,2pi)
    else:
        raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
    constellation = np.diag(np.exp(1j*phi))
    Modem.__init__(self, M, constellation,
     ↪  name='FSK',coherence=coherence.lower()) #set the base modem attributes

def demodulate(self, receivedSyms,coherence='coherent'):
    #overridden method in Modem class
    if coherence.lower()=='coherent':
        return self.iqDetector(receivedSyms)
    elif coherence.lower()=='noncoherent':
        return np.argmax(np.abs(receivedSyms),axis=1)
    else:
        raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
```

Класи, визначені у файлі modem.py, є лише логічними сутностями, які визначають поведінку (методи) та властивості (атрибути) різних модемів. Необхідний клас модему потрібно створити екземпляр, перш ніж його можна буде використовувати у подальшому програмуванні. Під час створення екземплярів класів модемів створюється об'єкт модему, який є сутністю зі специфічною характеристикою або функцією. Наступний код демонструє екземпляр об'єкта модему 16-PSK і те, як можна викликати його функції-члени для виконання модуляції та демодуляції [8]. Покажемо як можна реалізувати це в Python:

```
# modem.py

# Клас, що представляє загальні характеристики модема
class Modem:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def connect(self):
```

```
            raise NotImplementedError("Method connect() must be implemented in subclass.")

        def disconnect(self):
            raise NotImplementedError("Method disconnect() must be implemented in subclass.")

        def modulate(self, data):
            raise NotImplementedError("Method modulate() must be implemented in subclass.")

        def demodulate(self, signal):
            raise NotImplementedError("Method demodulate() must be implemented in subclass.")

    # Клас для реалізації модему 16-PSK
    class PSK16Modem(Modem):
        def modulate(self, data):
            print("Modulating data using 16-PSK modulation...")
            # Логіка модуляції 16-PSK

        def demodulate(self, signal):
            print("Demodulating signal using 16-PSK demodulation...")
            # Логіка демодуляції 16-PSK

    if __name__ == "__main__":
        # Приклад використання
        modem = PSK16Modem("Example", "16-PSK")
        modem.connect()
        modem.modulate("some_data")
```

```
modem.demodulate("some_signal")
modem.disconnect()
```

У цьому коді ми створили клас `PSK16Modem`, який успадковує функціональність загального класу `Modem`. У цьому підкласі ми реалізуємо методи `modulate()` та `demodulate()` специфічно для модему 16-PSK. При використанні цих методів для екземпляра модема вони виконають відповідні операції модуляції та демодуляції сигналів [9].

```
>> import sys
>> sys.path.append('path_where_digicommpy_resides') #search path
>> from DigiCommPy.modem import PSKModem #import the PSKModem class from modem.py
>> M = 16 #16 points in the constellation
>> pskModem = PSKModem(M) #create a 16-PSK modem object
>> pskModem.plotConstellation() #plot ideal constellation for this modem
>> import numpy as np # for numerical computing
>> nSym = 10 #10 symbols as input to PSK modem
>> inputSyms = np.random.randint(low=0, high = M, size=nSym) # uniform random
 ↪  symbols from 0 to M-1
array([10, 14,  1,  0,  0,  1, 10,  0, 14,  5])
>> modulatedSyms = pskModem.modulate(inputSyms) #modulate
array([-0.5-0.5j, 0.5-0.5j , 0.65+0.27j, 0.707+0.j, 0.707+0.j, 0.65+0.27j,
 ↪  -0.5-0.5j,  0.707+0.j, 0.5-0.5j, -0.27+0.65j])
>> detectedSyms = pskModem.demodulate(modulatedSyms) #demodulate
array([10, 14,  1,  0,  0,  1, 10,  0, 14,  5], dtype=int64)
```

5.2 Програмне визначення продуктивності інфокомунікаційної системи

Оцінка продуктивності може включати в себе різні параметри, такі як пропускна здатність, затримка, BER (бітова помилка), SNR (співвідношення сигнал/шум) та інші.

Наведемо приклад лістингу коду Python для оцінки BER за допомогою моделювання каналу з шумом:

```
import numpy as np

def generate_data(num_bits):
    return np.random.randint(0, 2, num_bits)
```

```
def add_noise(signal, snr_dB):
    snr = 10 ** (snr_dB / 10.0)
    noise_power = 1 / snr
    noise = np.sqrt(noise_power) * np.random.randn(len(signal))
    return signal + noise

def calculate_ber(original_signal, received_signal):
    num_errors = np.sum(original_signal != received_signal)
    ber = num_errors / len(original_signal)
    return ber

# Параметри
num_bits = 10000
snr_dB = 10

# Генерація вихідних даних
original_data = generate_data(num_bits)

# Моделювання каналу з шумом
received_data = add_noise(original_data, snr_dB)

# Обчислення BER
ber = calculate_ber(original_data, received_data)
print("Bit Error Rate (BER):", ber)
```

Цей код генерує випадкові біти, додає шум до сигналу з визначеним співвідношенням сигнал/шум (SNR), а потім обчислює BER порівнянням вихідних бітів з отриманими. Ви можете використовувати цей код для оцінки BER вашої інфокомунікаційної системи з різними значеннями SNR [10].

Зауважимо, що це лише один з можливих підходів до оцінювання продуктивності інфокомунікаційної системи, і в подальшому можливо розширити цей код для включення інших параметрів оцінки продуктивності, таких як пропускна здатність та затримка.

5.3 Висновки до розділу 5

Дослідження моделей цифрових модемів мобільних систем дозволяє краще зрозуміти вимоги сучасних комунікаційних потреб і визначити, як вони впливають на роботу модемів. Результати досліджень дозволяють вдосконалити роботу цифрових модемів в мобільних системах, щоб вони краще відповідали поточним вимогам.

Аналіз моделей дозволяє ідентифікувати можливі шляхи підвищення продуктивності цифрових модемів у мобільних системах. Розуміння роботи модемів дозволяє розробляти та вдосконалювати алгоритми з метою забезпечення кращої відповідності їхньої роботи вимогам сучасних комунікаційних стандартів. Знання про роботу цифрових модемів дозволяє їм краще адаптуватися до змін в мобільних системах та швидше впроваджувати нові функції і можливості.

Отже, дослідження моделей цифрових модемів мобільних систем допомагає вдосконалити їх роботу, підвищити продуктивність та відповідність вимогам сучасних комунікаційних потреб.

6 ОХОРОНА ПРАЦІ

Під час розробки програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи на працівника, впливають такі небезпечні та шкідливі виробничі фактори [13]: підвищена запиленість та загазованість повітря робочої зони; підвищена та понижена температура повітря робочої зони; підвищений рівень шуму; підвищений рівень статичної електрики; підвищена напруженість електричного поля; недостатня освітленість повітря робочої зони; фізичні перевантаження (статичні); нервово - психічні перевантаження (перенапруга аналізаторів).

Відповідно до визначених факторів формуємо рекомендації щодо покращення умов праці на робочому місці.

6.1 Технічні рішення щодо безпечного виконання роботи

Робоче місце розробника програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи знаходиться в кабінеті, обладнаному офісними меблями, ПК та іншою оргтехнікою.

На робочому місці користувача ПК виникають небезпечні та шкідливі фактори: підвищений рівень шуму, несприятливі мікрокліматичні умови, недостатній рівень освітленості, шкідливі речовини, підвищений рівень електромагнітних випромінювань радіочастот, висока напруга електричної мережі, статична електрика та інші. Робота з ПК супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи.

Проектування робочих місць, забезпечених ПК, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця проектувальника повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що входить до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення [14].

Головними елементами робочого місця проектувальника є стіл і крісло. Основним робочим положенням є положення сидячи. Робоча поза сидячи викликає мінімальне стомлення працівника. Раціональне планування робочого місця передбачає чіткий порядок і постійність розміщення предметів, засобів праці і документації. Те, що потрібне для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Висота робочої поверхні столу для користувачів повинна регулюватися в межах 680-800 мм, при відсутності такої можливості висота робочої поверхні столу повинна бути 725 мм. Модульними розмірами робочої поверхні столу для ПК, на підставі яких повинні розраховуватися конструктивні розміри, слід вважати: ширину 800, 1200, 1400 мм, глибину 800 і 1000 мм при нерегульованої висоті, що дорівнює 725 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною – не менше 500 мм, глибиною на рівні колін - не менше 450 мм і на рівні простягнутої ноги – не менше 650 мм. Робочий стілець (крісло) повинен бути підйомно-поворотним і регульованим по висоті і кутам нахилу сидіння і спинки, а також - відстані спинки до переднього краю сидіння. Робоче місце необхідно обладнати підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах до 150 мм і по куту нахилу опорної поверхні підставки до 20 градусів. Підставка повинна мати рифлену поверхню і бортик по передньому краю заввишки 10 мм. Клавіатуру

слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача, або на спеціальній регульованій по висоті робочої поверхні, відокремленої від основної стільниці.

Електричний струм – являє собою прихований тип небезпеки, бо його важко визначити в токо- та неструмоведучих частинах устаткування, які є хорошими провідниками електрики. Смертельно небезпечним для життя людини вважають струм, величина якого перевищує 0,05 А, струм менше 0,05 А - безпечний (до 1000 В). З метою попередження уражень електричним струмом до роботи повинні допускатися тільки особи, що добре вивчили основні правила з безпеки виконання роботи.

Приміщення, де експлуатуються ПК, належать до приміщень без підвищеної небезпеки ураження людини електричним струмом. Вимоги електробезпеки і пожежної безпеки у приміщеннях, де встановлені ПК і все устаткування для обслуговування, ремонту та налагодження роботи їх, електропроводи і кабелі мають відповідати електробезпеці зони та мати апаратуру захисту від струму короткого замикання.

Лінії електромережі ПК, у приміщенні виконана як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників (заземлення або занулення), причому площі перерізу нульового робочого і нульового захисного провідника повинні бути не менші за площу перерізу фазового провідника.

Відповідно до правил електробезпеки в службовому приміщенні повинен здійснюватись постійний контроль стану електропроводки, запобіжних щитів, шнурів, за допомогою яких включаються в електромережу комп'ютери, освітлювальні прилади, інші електроприлади. Електричні установки, до яких відноситься практично все обладнання ПК, представляють для людини велику потенційну небезпеку, тому що в процесі експлуатації або проведенні профілактичних робіт людина може торкнутися частин, що знаходяться під напругою. Специфічна небезпека електроустановок - струмоведучі провідники, корпуси стійок ЕОМ і іншого устаткування, яка під напругою в результаті

пошкодження (пробою) ізоляції, не подають будь-яких сигналів, які попереджають людину про небезпеку. Реакція людини на електричний струм виникає лише при протіканні останнього через тіло людини. Виключно важливе значення для запобігання електротравматизму має правильна організація обслуговування діючих електроустановок ВЦ, проведення ремонтних, монтажних і профілактичних робіт.

Оскільки в приміщенні використовується понад п’ять ПК, тому на помітному місці встановлено аварійний резервний вимикач, який в разі небезпеки повністю знеструмлює електричну мережу (крім освітлення). В такому випадку при використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволено прокладати їх без металевих труб та гнучких металевих рукавів, що ми і спостерігаємо у приміщенні.

6.2 Технічні рішення з гігієни праці та виробничої санітарії

6.2.1 Мікроклімат

Згідно ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [16] мікроклімат виробничих приміщень – умови внутрішнього середовища цих приміщень, що впливають на тепловий обмін працюючих з оточенням шляхом конвекції, кондукції, теплового випромінювання та випаровування вологи. Ці умови визначаються поєднанням температури, відносної вологості та швидкості руху повітря, температури оточуючих людину поверхонь та інтенсивністю теплового (інфрачервоного) опромінення.

Мікроклімат виробничих приміщень нормується в залежності від теплових характеристик виробничого приміщення, категорії робіт по важкості і періоду року. Категорія виконуваних робіт під час розроблення програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи - 1а [17] (табл. 6.1).

Таблиця 6.1 – Параметри мікроклімату

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	21 ... 25 ° С 40 ... 60% до 0,1 м / с
Теплий	Температура повітря в приміщенні Відносна вологість Швидкість руху повітря	22 ... 28 ° С 40 ... 60% 0,1 ... 0,2 м / с

Для підтримання у виробничих приміщеннях метеорологічних умов, які задовольняють нормативні вимоги використовують систему вентиляції. Приміщення обладнано системою загально обмінної припливно-витяжної вентиляції. На кожну вентиляційну установку складений паспорт з технічною характеристикою та схемою установки.

Крім того, для підтримання температури в холодний період року використовують загальну систему опалення.

6.2.2 Склад повітря робочої зони

Важливе значення для нормальної життєдіяльності людини має чисте повітря певного хімічного складу: кисень 20,95%, азот 78,08%, інертні гази 0,97% (по об’єму). Але повітря такого складу не завжди є у виробничих приміщеннях, так як значна частина технологічних процесів супроводжується виділенням шкідливих речовин у вигляді газу, пари, пилу та аерозолів.

ГДК шкідливих речовин, які знаходяться в досліджуваному приміщені, наведені в таблиці 6.2.

Таблиця 6.2 – ГДК шкідливих речовин у повітрі

Назва речовини	ГДК, мг/м^3		Клас небезпечності
	Максимально разова	Середньо добова	
Оксид азоту	0,085	0,085	2
Вуглекислий газ	3	1	4
Пил нетоксичний	0,5	0,15	4
Озон	0,16	0,03	1

Під час роботи на ПК важливо, щоб повітря мало певний іонний склад. Рівні позитивних і негативних іонів у повітрі приміщень з ПК мають відповідати санітарно-гігієнічним нормам (табл. 6.3).

Таблиця 6.3 – Рівні іонізації повітря приміщень при роботі на ПК

Рівні	Кількість іонів в 1 см^3	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально необхідні	50000	50000

Забезпечення складу повітря робочої зони здійснюється за допомогою системи кондиціювання, регулярного провітрювання, та вологого прибирання.

6.2.3 Виробниче освітлення

Природне освітлення на робочому місці проектувальника є бічне одностороннє.

Сучасні норми визначають, що мінімальна освітленість встановлюється за характеристикою зорової роботи з найменшим розміром об'єкта розрізнення, контрастом об'єкта із фоном і характеристикою фону.

Норми освітленості при штучному освітленні та КПО (для III пояса світлового клімату) при природному та сумісному освітленні (характеристика зорової роботи – дуже високої точності) зазначені у таблиці 6.4:

Таблиця 6.4 - Норми освітленості в приміщенні

Характеристика зорової роботи	Найменший розмір об'єкта розрізнювання	Розряд зорової роботи	Підрозряд зорової роботи	Контраст об'єкта розрізнення з фоном	Характеристика фона	Освітленість, лк		КПО, $e_н$, %			
						Штучне освітлення		Природнє освітлення		Сумісне освітлення	
						Комбіноване	Загальне	Верхнє або верхнє	Бокове	Верхнє або верхнє і бокове	Бокове
Дуже високої точності	Від 0,15 до 0,3	II	г	великий	світлий	1000	300	7	2,5	4,2	1,5

Для максимального використання природного освітлення в приміщенні слід систематично очищувати вікна від пилу та встановити жалюзі. Віконні прорізи не затемнюються іншими будівлями.

Як джерела світла для штучного освітлення в приміщенні застосовуються люмінесцентні лампи типу ЛБ. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

6.2.4 Виробничий шум

Рівні шуму на робочих місцях визначаються за ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [18] (табл. 6.5).

Рівень шуму на робочих місцях не має перевищувати 60 дБА, що досягається застосуванням малошумного обладнання, використанням спеціальних матеріалів для обшивки приміщень, а також різноманітними звукопоглинальними пристроями (перегородки, кожухи, прокладки тощо).

Таблиця 6.5 Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот

Вид трудової діяльності, робочі місця	Рівні звукового тиску в дБ в октавних смугах із середньогеометричними частотами, Гц									Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв.
	31,5	63	125	250	500	1000	2000	4000	8000	
Творча діяльність, обробка даних,	86	71	61	54	49	45	42	40	38	60

6.2.5 Виробничі випромінювання

Значення напруженості електростатичного поля на робочих місцях із ПК (як у зоні екрана дисплея, так і на поверхнях обладнання, клавіатури, друкувального пристрою) мають не перевищувати гранично допустимих [19] (табл. 6.6).

Таблиця 6.6 – Допустимі параметри електромагнітних випромінювань

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відеомонітору	10 В/м
Напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відеомонітору	0,3 А/м
Напруженість електростатичного поля не повинна перевищувати:	для дорослих користувачів 20кВ/м для дітей 15кВ/м

Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень [16].

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05 м від екрана та корпуса відео термінала при будь-яких положеннях регулювальних пристроїв не повинна перевищувати ОД бер/год (100 мкР/год).

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосовувати при екранні фільтри, локальні світлофільтри (засоби індивідуального захисту очей) та інші засоби захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат (згідно Директиви № 90/270/ЕЄС [20]).

6.3 Пожежна безпека

Пожежна безпека – стан об'єкта, при якому виключається можливість пожежі, а в разі його виникнення запобігається вплив на людей небезпечних факторів пожежі і забезпечується захист матеріальних цінностей. Пожежна безпека забезпечується системою запобігання пожежі і системою пожежного захисту.

Пожежна безпека приміщення повинна відповідати вимогам Кодексу цивільного захисту України [21] та «Правила пожежної безпеки в Україні» [22].

За вибуховою і пожежною небезпекою приміщення належить до категорії Д, згідно з нормами технологічного проектування «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [23]. У приміщенні знаходяться ПК та інша оргтехніка, що можуть спричинити пожежу.

Згідно даних наведених у таблиці 6.7 будівля, в якій знаходиться приміщення, має II ступінь вогнестійкості [24] приміщення можна віднести до категорії вибухопожежонебезпеки В (табл. 6.7).

Таблиця 6.7. – Визначення ступеня вогнестійкості будівельних конструкцій

Ступінь вогнестійкості будинків	Мінімальні межі вогнестійкості будівельних конструкцій (у хвилинах) та максимальні межі поширення вогню по них (см)								
	стіни				колони	сходові площадки, косоури, сходи, балки, марші сходових клітків	перекриття міжповерхові (у т. ч. горищні та над підвалами)	елементи суміщених покриттів	
	несучі та сходових клітків	самонесучі	зовнішні ненесучі	внутрішні ненесучі (перегородки)				плити, настили, прогони	балки, ферми, арки, рами
III	REI 120 M0	REI 60 M0	E15, M0 E30, M1	EI 15 M1	R 120 M0	R 60 M0	REI 45 M1	Не нормуються	

Клас приміщення і зон з вибухо- і пожежонебезпеки П-IIа (приміщення, у якому знаходяться тверді горючі речовини та матеріали.)

6.3.1 Технічні рішення системи запобігання пожежі

Для запобігання виникнення пожежі в приміщенні застосовують такі заходи:

- щорічне проведення повторних протипожежних інструктажів та занять за програмою пожежно-технічного мінімуму з особами, що відповідальні за пожежну безпеку;
- утримання в справному стані засобів протипожежного захисту;
- своєчасне інформування про несправність пожежної техніки, систем протипожежного захисту, водопостачання тощо.

Електромережі, електроприлади і апаратура експлуатується тільки у справному стані з урахуванням вказівок та рекомендацій підприємств-виготовлювачів. У разі виявлення пошкоджень електромереж, вимикачів,

розеток та інших електровиробів слід негайно вимкнути їх та вжити необхідних заходів щодо приведення в пожежобезпечний стан.

6.3.2 Технічні рішення системи протипожежного захисту

Протипожежний захист – це комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, запобігання пожежі, обмеження її розповсюдження, а також на створення умов для успішного гасіння пожежі.

У всіх службових приміщеннях обов’язково повинен бути «План евакуації людей при пожежі», що регламентує дії персоналу у разі виникнення вогнища загоряння і в якому зазначено місця розташування пожежної техніки. Як відомо пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання. У робочому приміщенні присутні всі три основні чинника, необхідні для виникнення пожежі. Горючими компонентами є: будівельні матеріали для акустичної і естетичної обробки приміщень, перегородки, двері, підлоги, ізоляція кабелів і ін

Джерелами запалювання у ВЦ можуть бути електронні схеми від ПК, прилади, застосовувані для технічного обслуговування, пристрої електроживлення, кондиціонування повітря, де внаслідок різних порушень утворюються перегріті елементи, електричні іскри та дуги, здатні викликати загоряння горючих матеріалів. У сучасних ПК дуже висока щільність розміщення елементів електронних схем. У безпосередній близькості один від одного розташовуються сполучні дроти, кабелі. При протіканні по них електричного струму виділяється значна кількість теплоти. При цьому можливо оплавлення ізоляції. Для відведення надлишкової теплоти від ЕОМ служать системи вентиляції та кондиціонування повітря. При постійному дії ці системи представляють собою додаткову пожежну небезпеку.

При проведенні обслуговуючих, ремонтних і профілактичних робіт використовуються різні мастильні речовини, легкозаймисті рідини, прокладаються тимчасові електропровідниками, ведуть пайку та чистку окремих

вузлів. Виникає додаткова пожежна небезпека, яка потребує додаткових заходів пожежного захисту. Зокрема, при роботі з паяльником слід використовувати неспалену підставку з нескладними пристроями для зменшення споживаної потужності в неробочому стані.

Однією з найбільш важливих завдань пожежної захисту є захист будівельних приміщень від руйнувань та забезпечення їх достатньої міцності в умовах впливу високих температур при пожежі. З огляду на високу вартість електронного устаткування приміщення, а також категорію його пожежної небезпеки, будинки для ВЦ і частини будинку іншого призначення, в яких передбачено розміщення ПК повинні бути 1 і 2 ступеня вогнестійкості.

Для виготовлення будівельних конструкцій використовуються, як правило, цегла, залізобетон, скло, метал та інші негорючі матеріали. Застосування дерева повинна бути обмежено, а в разі використання необхідно просочувати його вогнезахисними складами.

До засобів гасіння пожежі, призначених для локалізації невеликих загорянь, відносяться внутрішні пожежні водопроводи, вогнегасники, сухий пісок, азбестові ковдри і т. д. Застосування води в машинних залах ПК, сховищах носіїв інформації, приміщеннях контрольно-вимірювальних приладів, зважаючи на небезпеку пошкодження або повного виходу з ладу дорогого устаткування можливо у виняткових випадках, коли пожежа приймає загрозливо великі розміри. При цьому кількість води повинна бути мінімальною, а пристрої ПК необхідно захистити від попадання води, накриваючи їх брезентом або полотном.

Для гасіння пожеж на початкових стадіях широко застосовуються вогнегасники. У приміщенні застосовуються головним чином вуглекислотні вогнегасники, перевагами якого є висока ефективність гасіння пожежі, схоронність електронного устаткування, діелектричні властивості вуглекислого газу, що дозволяє використовувати ці вогнегасники навіть у тому випадку, коли не вдається знеструмити електроустановку відразу.

ВИСНОВКИ

Виконане дослідження програмного додатку для оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи допомагає вдосконалювати та оптимізувати роботу системи, що є ключовим для забезпечення ефективного та надійного безпровідного зв'язку.

Практичне значення полягає в можливості оцінювати та аналізувати роботу цифрового приймача в реальних умовах зв'язку. Це дозволяє виявляти проблеми та недоліки в системі, що можуть впливати на якість зв'язку, і вчасно їх виправляти. Завдяки програмному додатку можна тестувати різні функції та алгоритми, що впливають на роботу цифрового приймача. Це дозволяє розробникам вдосконалювати та доповнювати функціональні можливості системи.

Практичне значення полягає в можливості проведення тестів під різними умовами зв'язку, такими як різні рівні шуму, дисторсії, допуску до помилок, щоб визначити, як система поводиться в реальних умовах експлуатації. Програмний додаток дозволяє визначати оптимальні налаштування та параметри роботи цифрового приймача для оптимального використання ресурсів, таких як енергія, обчислювальна потужність та пропускна здатність каналу передачі даних. Практичне значення полягає в можливості використовувати програмний додаток для тестування та валідації роботи цифрового приймача з використанням нових протоколів та стандартів безпровідних комунікацій.

Дослідження програмних моделей цифрових приймачів інфокомунікаційних систем дозволило здійснити симуляцію та аналіз різних аспектів їхньої роботи. Використання програмних моделей дозволяє ефективно вивчати та вдосконалювати цифрові приймачі без необхідності в експериментальних установках.

Аналіз моделей цифрових приймачів для фазової маніпуляції сигналів показав їхню ефективність та надійність в різних умовах та сценаріях.

Оптимізація таких моделей може покращити їх продуктивність та забезпечити кращу роботу в реальних умовах.

Моделювання цифрових приймачів для частотної маніпуляції сигналів, зокрема FSK, дозволяє проводити аналіз та вдосконалення їхньої роботи з метою забезпечення ефективності та надійності зв'язку. Програмна оптимізація цифрових приймачів дозволяє досягнути їхньої оптимальної продуктивності у програмних додатках, що є важливим аспектом для забезпечення швидкодії та ефективності роботи інфокомунікаційних систем. Дослідження моделей цифрових модемів мобільних систем виявило їхню важливість для покращення роботи в мобільних системах та відповідності вимогам сучасних комунікаційних потреб. Вдосконалення таких моделей може покращити якість зв'язку та забезпечити кращу роботу мобільних систем в умовах змінних чинників.

Вказані висновки підкреслюють важливість дослідження та вдосконалення цифрових приймачів і модемів для забезпечення надійної та ефективної роботи інфокомунікаційних систем у сучасному світі.

Розглянуті питання з охорони праці важливі для забезпечення безпеки та здоров'я працівників, які займаються розгортанням, обслуговуванням та експлуатацією мобільних телекомунікаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Frenzel, L.E. (2016). Principles of Electronic Communication Systems. New York: McGraw- Hill Education.

2. Quadrature Amplitude Modulation (QAM) (2023). What is it? Electrical4U. https://www.electrical4u.com/quadrature- amplitude- modulation- qam/ (accessed 14 February 2023).

3. Beard, C., Stallings, W., and Tahiliani, M.P. (2015). Wireless Communication Networks and Systems, Global Edition. Pearson.

4. Salvi, S. and Geetha, V. (2019). From light to Li-F i: research challenges in modulation, MIMO, deployment strategies and handover. International Conference on Data Science and Engineering (ICDSE), Patna, India (26–28 September 2019), pp. 107–119.

5. Васильківський, М. В. Програмні технології в інфокомунікаційних системах. Навчальний посібник для студентів спеціальності 172 «Телекомунікації та радіотехніка» : електронний навчальний посіб-ник комбінованого (локального та мережного) використання [Електронний ресурс] / Васильківський М. В., Бортник Г. Г., Кичак В. М. – Вінниця : ВНТУ, 2023. – 141 с.

6. Основи програмування (Python, Java) : лабораторний практикум / Смотр О., Придатко О., Малець І. – Львів : ЛДУ БЖД, 2019. – 134 с.

7. Програмування мовою Python / О.М. Васильєв. — Тернопіль: Навчальна книга – Богдан, 2019. — 504 с.; іл.

8. Васильківський, М., Коломієць, А., & Грабчак, Н. (2022). Дослідження функціональних параметрів інфокомунікаційних мереж 6G. Вісник Хмельницького національного університету, (6), 46–52. https://www.doi.org/10.31891/2307-5732-2022-315-6-46-52

9. Шарадкін Д.М., Субач І.Ю., Микитюк А.В. Інструментальні засоби Python для моделювання та си-стемного аналізу часових рядів при вирішенні задач кіберзахисту інформаційно-комунікаційних си-стем: навч. пос. / Шарадкін Д.М.,

Субач І.Ю., Микитюк А.В.; ІСЗЗІ КПІ ім. Ігоря Сікорського. – Київ : КПІ ім. Ігоря Сікорського, 2023.- 139 с.

10. Березовський В. Є. Чисельні методи з прикладами реалізації мовою Python / В. Є. Березовський, Л. Є.Ковальов, М. О.Медведєва : навчальний посібник. Умань : ВПЦ «Візаві», 2023. 88 с.

11. Васильківський, М., Прикмета, А., Олійник, А., & Нікітович, Д. (2023). Оптимізація інтелектуальних телекомунікаційних мереж. Вісник Хмельницького національного університету, Технічні науки. – 2023. – № 1. (317). – С. 33–41. doi: 10.31891/2307-5732-2023-317-1-33-41

12. М. Васильківський, О. Городецька, Б. Климчук, і В. Говорун, «Стратегії технологічного розвитку апаратного забезпечення інфокомунікаційних радіомереж», ІТКІ, вип. 56, вип. 1, с. 83–91, Бер 2023.

13. ДСТУ OHSAS 18002:2015. Системи управління гігієною та безпекою праці. Основні принципи виконання вимог OHSAS 18001:2007 (OHSAS 18002:2008, IDT). К. : ГП «УкрНИУЦ», 2016. 21 с

14. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. URL: http://sop.zp.ua/norm_npaop_0 _ 00-7_ 15-18_01_ua.php

15. ДБНВ.2.5-27-2006. Захисні заходи електробезпеки в електроустановках будинків і споруд. К. : Мінбуд України, 2006. 154 с

16. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. - [Електронний ресурс] - Режим доступу: http://mozdocs.kiev.ua/view.php?id=1972

17. Гігієнічна класифікація праці (за показниками шкідливості і небезпеки факторів виробничого середовища від 12.08.1986 № 4137-86. - [Електронний ресурс] - Режим доступу: http://zakon4.rada.gov.ua/laws/show/v4137400-86

18. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. - [Електронний ресурс] - Режим доступу: http://document.ua/sanitarni-normi-virobnichogo-shumu-ultrazvuku-ta-infrazvuku-nor4878.html

19. ДСНіП3.3.6.096-2002. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів. URL: http://zakon2.rada.gov.ua/laws/show/z0203-03

20. Директива № 90/270/ЕЄС «Про мінімум вимог безпеки і гігієни праці при роботі з екранними пристроями - [Електронний ресурс] - Режим доступу: https://osha.europa.eu/en/legislation/directives/provisions-on-workload-ergonomical-and-psychosocial-risks/osh-directives/5

21. Кодекс цивільного захисту України від 02.10.2012 № 5403-VI

22. НАПБА.01.001-14. Правила пожежної безпеки в Україні. К. : МВС Ук раїни, 2014. 47 с.

23. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпек. URL: https://dbn.co.ua/load/normativy/dstu/dstu_b_v_1_1_36/5-1-0-1759.

24. ДБН В.1.1-7:2016 Пожежна безпека об`єктів будівництва. Загальні ви моги. URL: http://www.poliplast.ua/doc/dbn _v.1.1-7-200

ДОДАТКИ

Додаток А

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ОЦІНЮВАННЯ РОБОТИ ЦИФРОВОГО ПРИЙМАЧА ПРОГРАМНО-КЕРОВАНОЇ ІНФОКОМУНІКАЦІЙНОЇ СИСТЕМИ

назва бакалаврської дипломної роботи

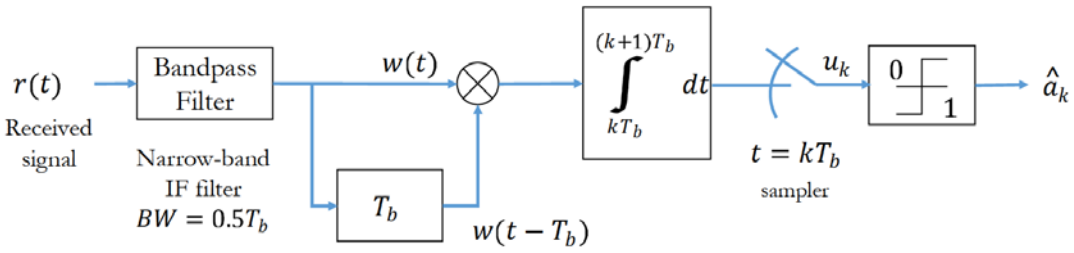

Рисунок 1 – Структурна схема субоптимального приймача DBPSK сигналів

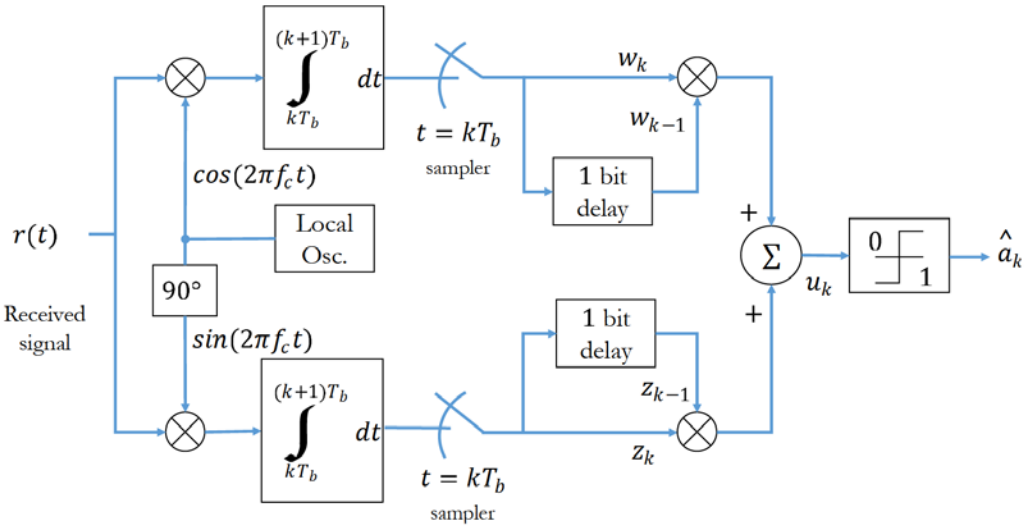

Рисунок 2 – Структурна модель оптимального некогерентного приймача DBPSK сигналів

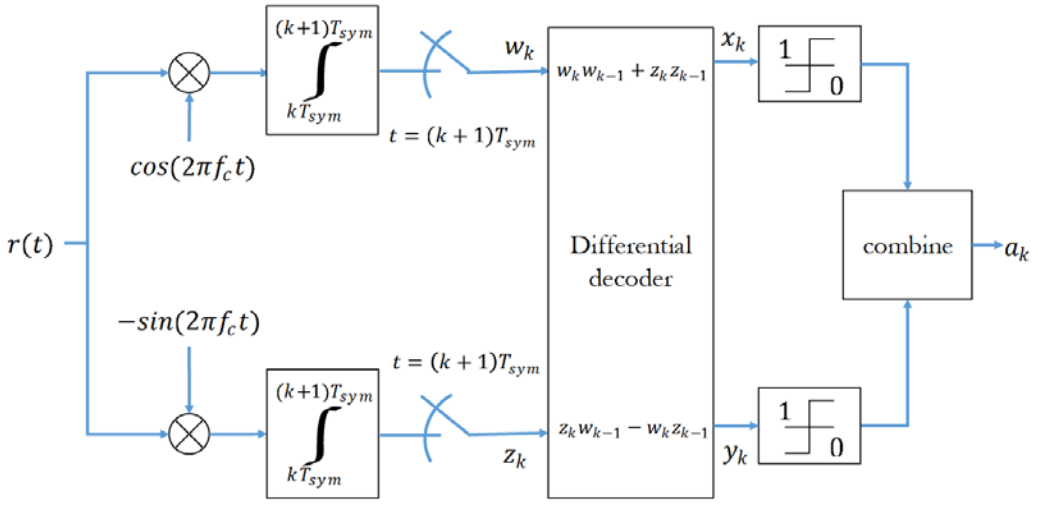

Рисунок 3 – Диференційний демодулятор базової смуги для π/4-DQPSK

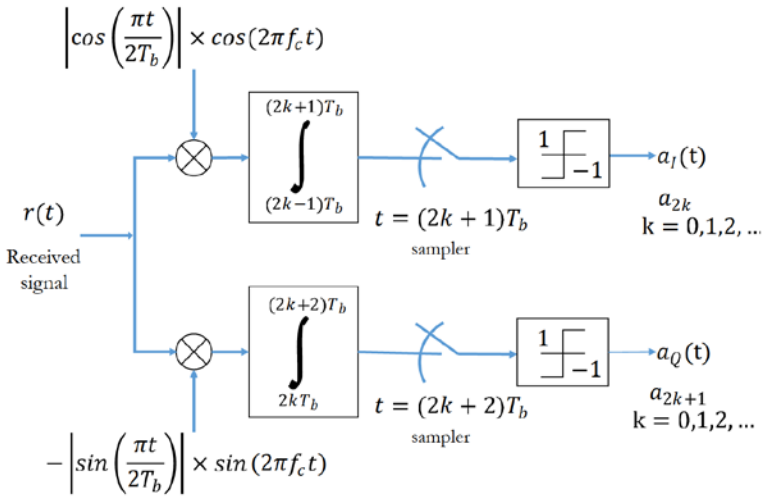

Рисунок 4 - Демодулятор MSK з використанням підходу комплексного представлення

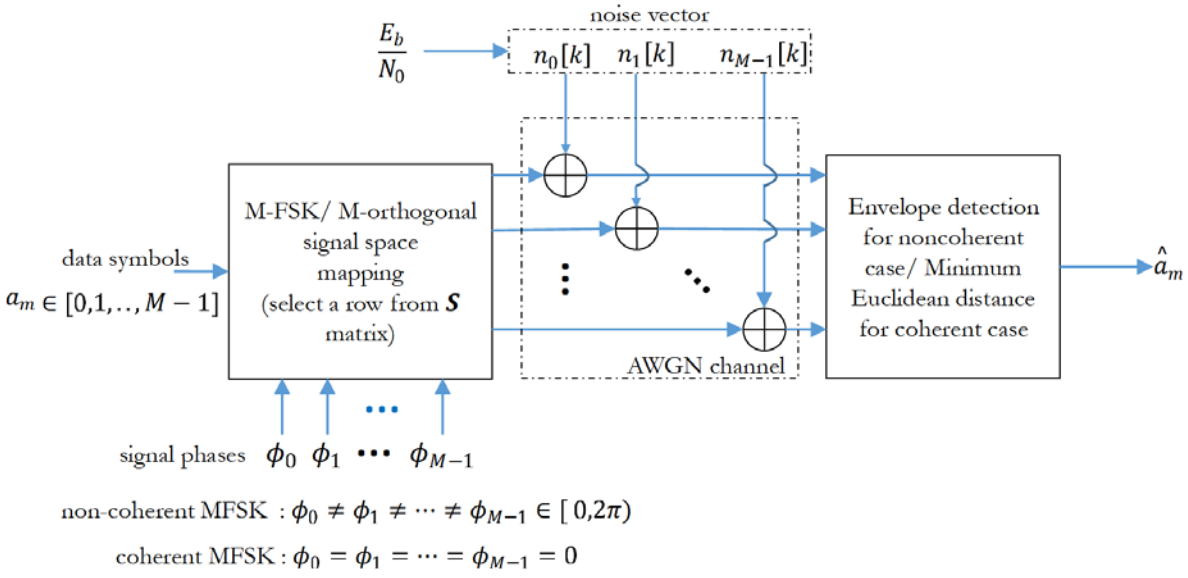

Рисунок 5 - Імітаційна модель для когерентного та некогерентного детектування MFSK сигналів

Рисунок 6 - Узагальнена архітектура традиційного RAN та O-RAN

Лістинг програми

```
import numpy as np
import abc
import matplotlib.pyplot as plt

class Modem:
    __metadata__ = abc.ABCMeta
    # Base class: Modem
    # Attribute definitions:
    #    self.M : number of points in the MPSK constellation
    #    self.name: name of the modem : PSK, QAM, PAM, FSK
    #    self.constellation : reference constellation
    #    self.coherence : only for 'coherent' or 'noncoherent' FSK
    def __init__(self,M,constellation,name,coherence=None): #constructor
        if (M<2) or ((M & (M -1))!=0): #if M not a power of 2
            raise ValueError('M should be a power of 2')
        if name.lower()=='fsk':
           if (coherence.lower()=='coherent') or (coherence.lower()=='noncoherent'):
                self.coherence = coherence
            else:
              raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
        else:
            self.coherence = None
        self.M = M # number of points in the constellation
        self.name = name # name of the modem : PSK, QAM, PAM, FSK
        self.constellation = constellation # reference constellation

    def plotConstellation(self):
        """
        Plot the reference constellation points for the selected modem
        """
        from math import log2
        if self.name.lower()=='fsk':
            return 0 # FSK is multi-dimensional difficult to visualize

    fig, axs = plt.subplots(1, 1)
    axs.plot(np.real(self.constellation),np.imag(self.constellation),'o')
    for i in range(0,self.M):
        axs.annotate("{0:0{1}b}".format(i,int(log2(self.M))),
        ↪  (np.real(self.constellation[i]),np.imag(self.constellation[i])))

    axs.set_title('Constellation');
    axs.set_xlabel('I');axs.set_ylabel('Q');fig.show()

def modulate(self,inputSymbols):
    """
        Modulate a vector of input symbols (numpy array format) using the chosen
         modem. The input symbols take integer values in the range 0 to M-1.
    """
    if isinstance(inputSymbols,list):
        inputSymbols = np.array(inputSymbols)

    if  not (0 <= inputSymbols.all() <= self.M-1):
       raise ValueError('Values for inputSymbols are beyond the range 0 to M-1')

    modulatedVec = self.constellation[inputSymbols]
    return modulatedVec #return modulated vector

def demodulate(self,receivedSyms):
    """
      Demodulate a vector of received symbols using the chosen modem.
    """
    if isinstance(receivedSyms,list):
        receivedSyms = np.array(receivedSyms)

    detectedSyms= self.iqDetector(receivedSyms)
    return detectedSyms

def iqDetector(self,receivedSyms):
    """
    Optimum Detector for 2-dim. signals (ex: MQAM,MPSK,MPAM) in IQ Plane
    Note: MPAM/BPSK are one dimensional modulations. The same function can be
    applied for these modulations since quadrature is zero (Q=0)

    The function computes the pair-wise Euclidean distance of each point in the
    received vector against every point in the reference constellation. It then
    returns the symbols from the reference constellation that provide the
    minimum Euclidean distance.

    Parameters:
        receivedSyms : received symbol vector of complex form
    Returns:
        detectedSyms:decoded symbols that provide the minimum Euclidean distance

    """
```

```
        from scipy.spatial.distance import cdist
        # received vector and reference in cartesian form
        XA = np.column_stack((np.real(receivedSyms),np.imag(receivedSyms)))
       XB=np.column_stack((np.real(self.constellation),np.imag(self.constellation)))

        d = cdist(XA,XB,metric='euclidean') #compute pair-wise Euclidean distances
        detectedSyms = np.argmin(d,axis=1)#indices corresponding minimum Euclid.
         ↪ dist.
        return detectedSyms

class PAMModem(Modem):
    # Derived class: PAMModem
    def __init__(self, M):
        m = np.arange(0,M) #all information symbols m={0,1,...,M-1}
        constellation = 2*m+1-M + 1j*0  # reference constellation
        Modem.__init__(self, M, constellation, name='PAM') #set the modem attributes

class PSKModem(Modem):
    # Derived class: PSKModem
    def __init__(self, M):
        #Generate reference constellation
        m = np.arange(0,M) #all information symbols m={0,1,...,M-1}
        I = 1/np.sqrt(2)*np.cos(m/M*2*np.pi)
        Q = 1/np.sqrt(2)*np.sin(m/M*2*np.pi)
        constellation = I + 1j*Q #reference constellation
        Modem.__init__(self, M, constellation, name='PSK') #set the modem attributes

class QAMModem(Modem):
    # Derived class: QAMModem
    def __init__(self,M):
        if (M==1) or (np.mod(np.log2(M),2)!=0): # M not a even power of 2
            raise ValueError('Only square MQAM supported. M must be even power of 2')

        n = np.arange(0,M) # Sequential address from 0 to M-1 (1xM dimension)
        a = np.asarray([x^(x>>1) for x in n]) #convert linear addresses to Gray code
        D = np.sqrt(M).astype(int) #Dimension of K-Map - N x N matrix
        a = np.reshape(a,(D,D)) # NxN gray coded matrix
        oddRows=np.arange(start = 1, stop = D ,step=2) # identify alternate rows
        a[oddRows,:] = np.fliplr(a[oddRows,:]) #Flip rows - KMap representation
        nGray=np.reshape(a,(M)) # reshape to 1xM - Gray code walk on KMap

        #Construction of ideal M-QAM constellation from sqrt(M)-PAM
        (x,y)=np.divmod(nGray,D) #element-wise quotient and remainder
        Ax=2*x+1-D # PAM Amplitudes 2d+1-D - real axis
        Ay=2*y+1-D # PAM Amplitudes 2d+1-D - imag axis
        constellation = Ax+1j*Ay
        Modem.__init__(self, M, constellation, name='QAM') #set the modem attributes

class FSKModem(Modem):
    # Derivied class: FSKModem

    def __init__(self,M,coherence='coherent'):
        if coherence.lower()=='coherent':
            phi= np.zeros(M) # phase=0 for coherent detection
        elif coherence.lower()=='noncoherent':
            phi = 2*np.pi*np.random.rand(M) # M random phases in the (0,2pi)
        else:
            raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
        constellation = np.diag(np.exp(1j*phi))
        Modem.__init__(self, M, constellation,
         ↪ name='FSK',coherence=coherence.lower()) #set the base modem attributes

    def demodulate(self, receivedSyms,coherence='coherent'):
        #overridden method in Modem class
        if coherence.lower()=='coherent':
            return self.iqDetector(receivedSyms)
        elif coherence.lower()=='noncoherent':
            return np.argmax(np.abs(receivedSyms),axis=1)
        else:
            raise ValueError('Coherence must be \'coherent\' or \'noncoherent\'')
```

Додаток Б

(обов’язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного додатку оцінювання роботи цифрового приймача програмно-керованої інфокомунікаційної системи

Тип роботи: Бакалаврська дипломна робота
(БДР)

Підрозділ кафедра інфокомунікаційних систем і технологій, факультет інформаційних електронних систем
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,04 % Схожість 4,96 %

Аналіз звіту подібності (відмітити потрібне):

■ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа відповідальна за перевірку _____ Васильківський М.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Муревич Р.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Бортник С.Г.
(підпис) (прізвище, ініціали)